REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE IBN KHALDOUN - TIARET

MEMOIRE

Présenté à :

FACULTÉ MATHEMATIQUES ET INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

Pour l'obtention du diplôme de :

MASTER

Spécialité : Génie Logiciel

Par :

BOUSSAFI Yasmine

Sur le thème

Developing Sequence-Aware Recommender Systems with Hidden Markov Models

Soutenu publiquement le 20 /09/ 2021 à Tiaret devant le jury composé de :

Mr. Kouadria Abderrahmane	Grade	Maître assistant	Président
Mr. Boudaa Boudjemaa	Grade	Maître de conférences	Encadreur
Mme. Hamdani abdia	Grade	Maître assistant	Examinateur

2020-2021

This document is written by LaTeX

September 2021

Acknowledgment

First of all, I want to thank the Almighty and Merciful God, who has given me the strength and patience to do this modest work.

Secondly, I would like to thank my supervisor Mr. Boudaa Boudjemaa, whose office door was always open. and guided me in the right direction whenever I needed it. It was a great privilege and honor to work and study under his guidance.

My sincere thanks also goes to the jury for the interest they took in my thesis by agreeing to examine my work and enrich it with their insights.

And I would like to thank my family especially my loving and caring Mom whose words of encouragement and push for tenacity ring in my ears.

I would like to recognize the invaluable companionship that my loved one has provided me with, for their unconditional love, prayers, caring, and the comfort I found in them in the difficult times, for all the late nights and early mornings, and for keeping me In an excellent shape all the time. Thank you for being with me, I owe you everything.

Finally, I would also like to thank all the people and my friends who have supported me throughout the process. I will always appreciate all they have done.

BOUSSAFI YASSMINE

Dedication

I dedicate my dissertation work to my mother CHAIEB KHALDIA whose unconditional love and support inspires me to always do my best. She has such a giving heart, a selfless spirit and has served others throughout her life. Thank you for being my mom. I love you so much.

Table of Contents

Abstract

Recommender systems (RS) are one of the most successful applications of data mining and Machine learning technology in practice. It is generally based on the matrix completion problem formulation, where for each user-item pair only one interaction (rating) is considered. In many application domains, multiple user-item interactions of different types can be recorded over time. And a number of recent works have shown that this information can be used to build richer individual user models and to discover additional behavioral patterns that can be leveraged in the recommendation process in what is called Sequence-Aware Recommender Systems (SARS). This thesis aims to highlight this new kind of Recommender systems (SARS). In addition, a Hidden Markov Model (HMM) to develop SARS is studied. Results of the experiments on the Last.fm dataset, are presented using a different set of HMM parameters.

Keywords: Recommender systems, Sequence-Aware Recommender Systems, Sequential Recommendation, Sequential data, Hidden Markov Model, Next-Item recommendation.

Résumé

Les systèmes de recommandation sont l'une des applications les plus réussies de l'exploration de données et de la technologie d'apprentissage automatique dans la pratique. Ils sont généralement basés sur la formulation du problème de complétion de la matrice, où pour chaque paire utilisateur-article, une seule interaction (évaluation) est prise en compte. Dans de nombreux domaines d'application, de multiples interactions utilisateur-article de différents types peuvent être enregistrées dans le temps. Un certain nombre de travaux récents ont montré que ces informations peuvent être utilisées pour construire des modèles d'utilisateurs individuels plus riches et pour découvrir des modèles comportementaux supplémentaires qui peuvent être exploités dans le processus de recommandation dans ce qu'on appelle les systèmes de recommandation conscients des séquences SARS . Cette thèse vise à mettre en évidence ce nouveau type de systèmes de recommandation SARS. En outre, un modèle de Markov caché HMM pour développer les SARS est étudié. Les résultats des expériences sur le jeu de données Last.fm, sont présentés en utilisant un ensemble différent de paramètres HMM.

Mots clés: Systèmes de recommandation, systèmes de recommandation conscients de la séquence, recommandation séquentielle, données séquentielles, modèle de Markov caché, recommandation du prochain élément.

List of Tables

List of Figures

Acronyms

AI Artificial Intelligence. 3, 19, 20, 24, 38

APIs Application programming interface. 19

CB Content-Based. 5–8

CF Collaborative Filtering. 1, 5–8

CNN Convolutional Neural Networks. 17

FSM finite state machines. 26

GNN Graph Neural Networks. 17

GUI graphical user interface. 46, 47

HMM Hidden Markov Model. IV, VI, VII, 1–3, 19, 27–31, 34, 35, 37–47, 50–53

IDE Integrated development environment. 47

MAE Mean absolute error. 9

MC Markov Chains. 3, 16, 26–28, 39, 46

ML Machine learning. VI, 1–3, 5, 20–25, 38, 46, 47, 51, 52

MLE maximum likelihood estimation. 22, 35

RNN Recurrent Neural Networks. 1, 17

RS Recommender systems. IV, VI, 1–10, 15, 16, 18, 39, 41, 52

SARS Sequence-Aware Recommender Systems. VI, VII, 1–4, 11–13, 15–18, 39, 41, 42, 51, 52

General introduction

Background

Recommender systems (RS) are systems that suggest items that may be of interest to users. Such systems are prominent components of many of today's modern web applications, Recommender systems are and have been an important research area since the appearance of the first papers on Collaborative Filtering in the 1990s. Over the last two decades, there has been much researches about developing new approaches to Recommender systems. The interest still remains high because it constitutes a problem-rich research area and because of the abundance of practical applications that help users deal with information overload and provide personalized recommendations, content and services to them.

Recommender systems has often been based on a matrix completion problem formulation where each interaction, i.e user-item pair, is considered separately. However, in many application domains, user-item interactions can be recorded over time. Recommender systems that use user-item interactions recorded over time are called Sequence-Aware Recommender Systems. A number of recent works have shown that taking sequential data into account can yield richer recommendations and discover additional behavioural patterns. Sequential interaction logs often contain useful information on both short-term user interests as well as long-term sequential patterns that can be central to the success of a recommender system.

Applying Machine learning techniques to Recommender systems has become an increasingly popular research in in sequence modeling tasks. Although many of deep learning's successes have been based on Recurrent Neural Networks in this kind of tasks. However, in recent years, Hidden Markov Model have shown to be successful and efficient in sequence modeling tasks.

Problem Statement

The Sequence-Aware Recommender Systems is unappreciated problem in the Machine learning and Recommender systems community. Many media Recommender systems rarely track the Id's of the users that visit their sites over a long period. While cookies and browser fingerprinting can provide some level of user recognizability, those technologies are often not reliable enough and raise privacy concerns. Even if tracking is possible, lots of users does not have a richer history for making recommendations. Browsing on such sites (as last.fm), is naturally a sequential task, therefore using a matrix completion formulation would neglect the sequential nature of the data. Using a Sequence-Aware Recommender Systems could potentially increase capturing the sequential behavior of a user. Using sequential user interactions is especially important when there are many new or anonymous users as no long-term historical data about the general tastes of these users is available.

Although Hidden Markov Model have shown good results in sequence modeling tasks in other domains such as Natural language processing (NLP), investigating their usage in Sequence-Aware Recommender Systems remains an open topic of research. The main research question of this thesis is thus formulated as:

*How can Hidden Markov Model be used in
Sequence-Aware Recommender Systems development?*

Delimitations

The major research question emphasizes the investigation of Sequence-Aware Recommender Systems. In this thesis, the domain is limited to items, particularly those common within media. Where the goal of a system is to recommend items

Approach

To answer the research question, a Sequence-Aware Recommender Systems is developed. The first item a user listen to can be considered as the initial observable in the user's sequence, then based on this initial input the model is queried for a recommendation that depends on all the previous observables. An HMM with change of parameters will be presented in this thesis.

Outline

This thesis is structured in three chapters, a general introduction and conclusion:

- ***General introduction***

  An initiation to Recommender systems and the background, problem statement, and Delimitations of the thesis.

- ***Chapter 1 Sequence-Aware Recommender Systems***

  A theoretical introduction to Recommender systems and it's categories, challenges and goals. Beside a general overview of Sequence-Aware Recommender Systems, it's general architecture, and approaches.

- ***Chapter 2 Hidden Markov Model***

  Essential context and background knowledge around Artificial Intelligence and Machine learning (development, types and challenges). The main goals of this chapter is to give a familiarization with the fundamental concepts of Hidden Markov Model (Markov Chains, approaches, solutions, types and general structure as an example).

- ***Chapter 3 Developing Sequence-Aware Recommender Systems with Hidden Markov Models***

  Highlights on the practical application of Hidden Markov Model in developing Sequence-Aware Recommender Systems, First, the proposed model architecture, The evaluation metrics used for the experiment and dataset used. At last, the comparison and discussion of the obtained results.

- ***General conclusion***

  This last part draws a summary of this work and suggestions for potential future work.

Chapter 1

Sequence-Aware Recommender Systems

1 Introduction

Recommender systems (RS) are software applications that support users in finding items of interest within larger collections of objects in a personalized way [24]. Today, such systems are used in a variety of application domains, including for example e-commerce or media streaming, and receiving automated recommendations of different forms has become a part of our daily online user experience. Internally, such systems analyze the past behavior of individual users or of a user community as a whole to detect patterns in the data. This chapter gives a general presentation on the Recommender systems, mainly their kind of Sequence-Aware Recommender Systems.

2 Recommender Systems

Recommender systems (RS) attempt to discover user preferences, and to learn about them in order to anticipate their needs. RS provides specific suggestions about items within a given domain, which may be considered of interest to the given active user [5].

Recommender systems are a filtering systems that suggests items of possible interest for a given user. The problem of recommendation can be seen as the problem of estimating the user rating for items not yet rated by the user. Items predicted with high rating for a given user can be offered by the system as a recommendation.

2.1 Recommender Systems categories

There are several categories of Recommender systems but they usually work with two kinds of data, user-item interactions, such as ratings or buying behavior, and the attribute information about the users and items such as textual profiles or relevant keywords. Methods that use the former are referred to as Collaborative Filtering methods, where as methods that use the latter are referred to as Content-Based recommender methods. Note that Content-Based systems also use the ratings matrices in most cases, although the model is usually focused on the ratings of a single user rather than those of all users. In knowledge-based Recommender systems, the recommendations are based on explicitly specified user requirements. Instead of using historical rating or buying data, external knowledge bases and constraints are used to create the recommendation. Some Recommender systems combine these different aspects to create hybrid systems [5]. Hybrid systems can combine the strengths of various types of Recommender systems to create techniques that can perform more robustly in a wide variety of settings.

2.1.1 Collaborative Filtering Models

Collaborative Filtering (CF) models use the collaborative power of the ratings provided by multiple users to make recommendations. The main challenge in designing CF methods is that the underlying ratings matrices are sparse [7].

There are two types of methods that are commonly used in CF, which are referred to as memory-based methods and model-based methods:

1. ***Memory-based methods:*** Memory-based methods are also referred to as neighborhood-based Collaborative Filtering algorithms. These were among the earliest Collaborative Filtering algorithms, in which the ratings of user-item combinations are predicted on the basis of their neighborhoods. These neighborhoods can be defined in one of two ways:

   - *User-based Collaborative Filtering:* is to determine users U_s, who are similar to a target user A, and recommend ratings for the unobserved ratings of A by computing weighted averages of the ratings of U_s [1].
   - *Item-based Collaborative Filtering:* In order to make the rating predictions for target item B by user A, the first step is to determine a set S of items that are most similar to target item B. The ratings in item set S, which are specified by A, are used to predict whether the user A will like item B [1].

   The advantages of memory-based techniques are that they are simple to implement and the resulting recommendations are often easy to explain.

2. ***Model-based methods:*** In model-based methods, Machine learning and data mining methods are used in the context of predictive models. In cases where the model is parameterized, the parameters of this model are learned within the context of an optimization.

2.1.2 Content-Based Recommender Systems

Content-Based (CB) RS use item information such as name, author, category, keywords to make recommendations [9]. Unlike CF models, CB models only use user feedback and do not use feedback from other users to make a recommendation. This makes the model to focus on the user's interests and to recommend unknown items based on the categories of items the user has liked in the past. For example, if a user gives a positive feedback for pop music, the recommendation model is likely to recommend another pop music because it has learns that the user likes this category of music. Unlike Collaborative Filtering methods, Content-Based models do not suffer from the cold-start item problem because some users have probably already interacted with items similar to the new item [9]. However, Content-Based models have some disadvantages:

- The provided recommendations lack diversity as they are always part of the same area of interest that the user likes.
- In order to make relevant recommendations, the user must give sufficient feedback. These types of methods are strongly impacted by the user's cold-start problem.

2.1.3 Knowledge-Based Recommender Systems

They are used to recommend infrequently purchased items such as cars, travel, financial services, etc [10]. In these cases, it is difficult to know the user's tastes and their preferences are likely to change between two interactions. For this type of system, it is the users who will explicitly give information about the item they are looking for. Then, the system will return the items according to the needs expressed by the user.

2.1.4 Utility-Based Recommender Systems

In utility-based Recommender systems, a utility function is defined on the product features in order to compute the probability of a user liking the item [11]. The central challenge in utility-based methods is in defining an appropriate utility function for the user at hand. It is noteworthy that all recommender schemes, whether collaborative, Content-Based, or knowledge-based methods, implicitly rank the recommended items on the basis of their perceived value (or utility) for the target user. In utility-based systems, this utility value is based on a function that is known a priori. In this sense, such functions can be viewed as a kind of external knowledge. Therefore, utility-based systems can be viewed as a specific case of knowledge-based Recommender systems.

2.1.5 Demographic Recommender Systems

Rely on user demographic information [12], such as age and country, to assign a class. Then, from a rule-based system, the recommendation is made based on the user's class. These methods are not very effective because using demographic information alone does not fully model the user tastes. Nevertheless, this information can be useful for creating hybrid models.

2.1.6 Hybrid Recommender Systems

Hybrid Recommender systems combine several types of models to improve performance and overcome weaknesses related to the type of Recommender systems used [8]. One of the most common combinations is to mix a Content-Based method with a Collaborative Filtering method. This reduces the cold start problem of Collaborative Filtering methods. Table 1.1 shows some of the combination methods that have been employed.

Hybridization method	**Description**
Weighted [13]	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation
Switching [14]	The system switches between recommendation techniques depending on the current situation.
Mixed [15]	Recommendations from several di¡erent recommenders are presented at the same time
Feature combination [16]	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade [17]	One recommender refines the recommendations given by another
Feature augmentation [18]	Output from one technique is used as an input feature to another.
Meta-level [19]	The model learned by one recommender is used as input to another.

Table 1.1: Hybridization methods

For instance, CHAMELEON[1], a deep Learning hybrid-based recommender system, uses the interaction between user and item, item content and user context to address the specific challenges of news recommendation. This architecture allows CHAMELEON to reduce the problem of cold-start.

[1]Gabriel de Souza Pereira Moreira. CHAMELEON: a deep learning meta-architecture for news recommender systems. In Proceedings of the 12th ACM Conference on Recommender Systems, RecSys 2018, Vancouver, BC, Canada, October 2-7, 2018, pages 578–583, 2018.12

2.2 Challenges of Recommender systems

In this section challenges [6] in Recommender systems will be brought to light:

- ***Cold Start Problem:*** This problem occurs when new users enter the system or new items are added to the catalog. In such cases, neither the taste of the new users can be predicted nor can the new items be rated or purchased by the users leading to less accurate recommendations.

- ***Synonymy:*** Arises when an item is represented with two or more different names or entries having similar meanings. In such cases, the recommender cannot identify whether the terms represent different items or the same item. To alleviate the problems of synonymy, different techniques including ontologies, the Single Value Decomposition (SVD) techniques, and Latent Semantic Indexing (LSI) could be used.

- ***Shilling Attacks:*** What happens if a malicious user or competitor enters into a system and starts giving false ratings on some items either to increase the item popularity or to decrease its popularity. Such attacks can break the trust on the recommender system as well as decrease the performance and quality of recommenders. More information about *shilling attacks* can be found in [20].

- ***Privacy:*** Feeding personal information to the Recommender systems results in better recommendation services but may lead to issues of data privacy and security. Users are reluctant to feed data into RS that suffer from data privacy issues. Therefore, a recommender system should build trust among their users,

- ***Limited Content Analysis and Overspecialization:*** Content-Based recommenders rely on content about items and users to be processed by information retrieval techniques. The limited availability of content leads to problems including overspecialization.

- ***Grey Sheep:*** Occurs in pure CF systems where opinions of a user do not match with any group and therefore, is unable to get benefit of recommendations. Pure CB filtering can resolve this issue where items are suggested by exploiting user personal profile and contents of items being recommended.

- ***Sparsity:*** The availability of huge size of data about items the catalog and the disinclination of users to rate items make a dispersed profile matrix leading to less accurate recommendations. The sparse rating in CF systems makes it difficult to make accurate predictions about items.

- ***Scalability:*** The rate of growth of nearest-neighbor algorithms shows a linear relation with number of items and number of users. It becomes difficult for a typical recommender to process such large-scale data.

- ***Latency Problem:*** Is when new items are added more frequently to the database, where the recommender suggests only the already rated items as the newly added items are not yet rated.
- ***Evaluation and the Availability of Online Datasets:*** Evaluating a recommender system determines among the other things, its quality. The design of evaluation criteria and selection of suitable evaluation metrics is a key problem in Recommender systems. Most traditional Recommender systems evaluate system results and algorithms by considering dataset divided into a test set and apply metrics like Mean absolute error (MAE), Precision, and F-Measure for evaluation.
- ***Context-Awareness:*** Aggregates all categories that represent the setting in which recommender is deployed, It is envisioned that the upcoming Recommender systems will use contextual information obtained through mobile services infrastructure and will include the user's short and long term history, location, entries in the calendar, and the information that the user provides to social networks. This can greatly affect the performance of recommender. Finding out user preferences and context-related information is the key to come up with relevant recommendations for the user.

2.3 Goals of Recommender Systems

Increasing product sales is the primary goal of a recommender system. RS are utilized by merchants to increase their profit. By recommending carefully selected items to users, RS bring relevant items to the attention of users. This increases the sales volume and profits for the merchant. Although the primary goal of a recommendation system is to increase revenue for the merchant [1], the common operational and technical goals of RS are as follows:

1. ***Accuracy:*** The most obvious operational goal of a recommender system is to recommend items that are relevant to the user at hand. Users are more likely to consume items they find interesting.
2. ***Novelty:*** RS are truly helpful when the recommended item is something that the user has not seen in the past. For example, popular music of a preferred genre would rarely be novel to the user. Repeated recommendation of popular items can also lead to reduction in sales diversity [2].
3. ***Serendipity:*** wherein the items recommended are somewhat unexpected, and therefore there is a modest element of lucky discovery, as opposed to obvious recommendations. Serendipity is different from novelty in that the recommendations are truly surprising to the user [3], rather than simply something they did not know about before.
4. ***Diversity:*** RS typically suggest a list of top-k items. When all these recommended items are very similar, it increases the risk that the user might not like any of these

items. On the other hand, when the recommended list contains items of different types, there is a greater chance that the user might like at least one of these items. Diversity has the benefit of ensuring that the user does not get bored by repeated recommendation of similar items.

Aside from these concrete goals, a number of soft goals are also met by the recommendation process both from the perspective of the user and merchant. From the perspective of the user, recommendations can help improve overall user satisfaction with the Web site. At the merchant end, the recommendation process can provide insights into the needs of the user and help customize the user experience further.

However, RS have expanded beyond the traditional domain of product recommendations [4].In the following, the table 1.2 shows Examples of real-world recommendation goals.

System	**Product Goal**
Amazon	Books and other products
Netflix	DVDs, Streaming Video
Jester	Jokes
GroupLens	News
MovieLens	Movies
last.fm	Music
Google News	News
Google Search	Advertisements
Facebook	Friends, Advertisements
Pandora	Music
YouTube	Online videos
Trip advisor	Travel products
IMDb	Movies

Table 1.2: Examples of products recommended by various real-world recommender systems

2.4 Sequential and classical recommender systems

The most common approach to generate recommendations is to discard any sequential information and learn what items a user is typically interested in. On the other hand, recommendations of sequential methods are based only on the last user events by learning what an arbitrary user buys next when he has bought a certain item in the recent past [22]. Both methods have their strengths and disadvantages.

3 Sequence-Aware Recommender Systems

A Sequence-Aware Recommender Systems (SARS) is a recommender system that models the dynamics in user's interaction by extracting of sequential features from the logs of historical user activity, with the ultimate goal of adapting to user's short-term needs and providing high quality suggestions on the following item(s) the user should interacts with [25].

3.1 Characterizing Sequence-Aware Recommender Systems

SARSs are different from the traditional matrix-completion setup in a number of ways [24]. Figure 1.1 [25] gives a high-level overview of the problem, its inputs, outputs, and specific computational tasks. In general, the ordering of the objects can be relevant both with respect to the inputs and to the outputs.

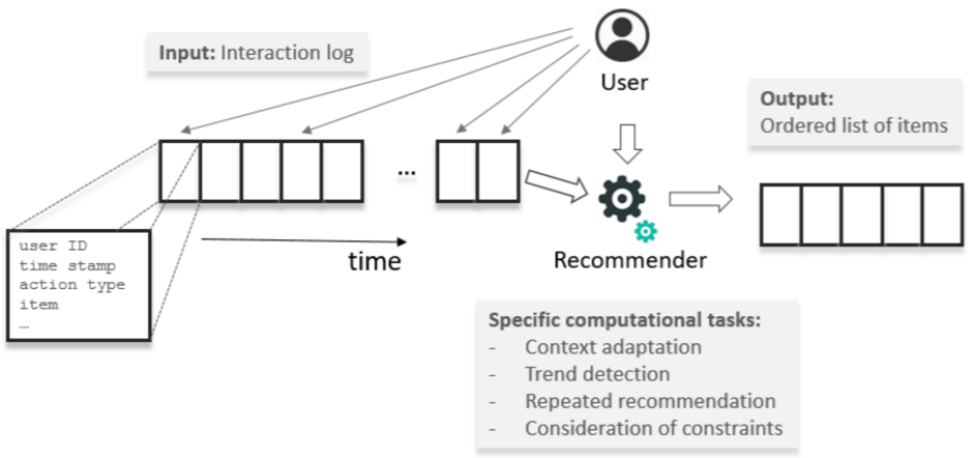


Figure 1.1: High-level Overview of Sequence-Aware Recommendation Problems.

- ***Inputs***: The main input to sequence-aware recommendation problems is an ordered and often timestamped list of past user actions. Users can be already known by the system or anonymous ones. Overall, the inputs can be considered as a sort of enriched click-stream data [24].
- ***Outputs***: The output of a sequence-aware recommender are ordered lists of items. They are similar to those of a traditional "item-ranking" recommendation setup. However, in some sequence-aware recommendation scenarios, the ordering of the objects in the recommendation list can be relevant as well. Instead of considering the list of recommendations as a set of alternatives for the user, there are scenarios where the user should consider all recommendations and do this in the provided order [24].

- ***Computational Tasks:*** This section identifies the four main goals (computational tasks) that can be achieved with the help of Sequence-Aware Recommender Systems in different application scenarios:
  - ***Context Adaptation:*** Making context-adapted recommendations from past interaction data is the main goal of Sequence-Aware Recommender Systems plus understanding the user's situation and goals. The following types can be differentiate depending on the availability of historical data for individual users [24].
    - ***Last-N interactions based recommendation***: In this scenario, only the last N user actions are considered.
    - ***Session-based recommendation***: In this scenario only the last sequence of actions of a user is known and this sequence of actions is limited to a period of time when the user interacted with the site.
    - ***Session-aware recommendation***: This scenario happens whene there is knowledge about the user's actions in the last session and about their past behavior, a Sequence-Aware Recommender Systems can be based on a combination of long term and short-term interest models.
  - ***Trend Detection:*** Sequence-Aware Recommender Systems have a secondary goal *trend detection* in a given sequence [24]. Based on sequential log information two types can be defined:
    - ***Individual trends:*** Changes in the interest in certain items can also happen at an individual level. These interest changes can be caused when there is a "natural" interest drift.
    - ***Community trends:*** Sequence-Aware Recommender Systems can aim to detect and utilize popularity patterns in the interaction logs to improve the recommendations considering the changing and the popularity of items in a user community.
  - ***Repeated Recommendation:*** The following categories of repeated recommendation scenarios [24] can be defined since recommending items that the user already knows or has purchased in the past can be meaningful:
    - ***Repeated recommendations as reminders:*** Repeated recommendations can help to remind users of things they found interesting in the past. Depending on the domain, these reminders could relate to objects that the user has potentially forgotten.
    - ***Identifying repeated user behavior patterns:*** SARS can use interaction logs to identify patterns of repeated user behavior. Repeated user actions are particularly relevant for app recommendation problems.

- ***Consideration of Order Constraints and Sequential Patterns:*** The characteristics of items may pose a sort of ordering between interactions, we distinguish between *strict* and *weak* order constraints, and order constrains that *mined* directly from the logs of user's activity [25] .
  - ***Strict or Weak order constraints:*** In the sequential recommendation there might be strict requirements regarding the order of different items that have to be considered by the recommender differing from Weak order constraints that does not pose any strict ordering between them.
  - ***Mined order constraints:*** SARS can mine such sequential patterns from the log data, e.g., to automatically detect that users who watched a certain movie later on have watched its sequel.

3.2 Sequential Recommendation Tasks

Figure 1.2 demonstrate two representative recommendation tasks [21] :

- **Next-Item Recommendation:** A user behavior contains only one item.
- **Next-Basket Recommendation:** A user behavior contains more than one items.

However, either in *Next-Item Recommendation* or *Next-Basket Recommendation*, the model is to predict the next item(s) for a user, and the most popular output is still a list of Top-K ranked items, where the ranking could be determined by probabilities, absolute scores or relative rankings.

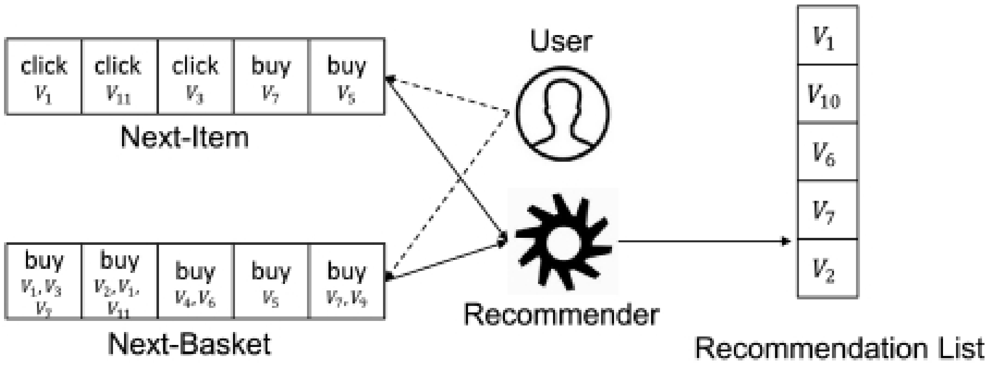


Figure 1.2: Next-Item and Next-Basket Recommendation

3.3 Behavior sequence types

In sequential data there are two types of behaviors [21]:

- ***Behavior object:*** Refers to the items or services that a user chooses to interact with.
- ***Behavior type:*** Refers to the way that a user interacts with items or services.

Given these concepts, a behavior can be considered as a combination of a behavior type and a behavior object, (a user interacting with a behavior object by a behavior type).

Figure 1.3 shows a schematic diagram of the sequential recommendation. c_i:behavior type, o_i:behavior object. A behavior a_i is represented by a 2-tuple, i.e., $a_i = (c_i, 0_i)$. A behavior sequence is a list of 2-tuples in the order of time.

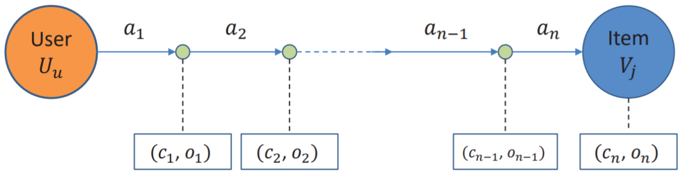


Figure 1.3: A behavior sequence (trajectory)

A sequential recommender system is referred to a system that takes a user's behavior trajectories as input and then adopts recommendation algorithms to recommend appropriate items or services to the user. The input behavior sequence is polymorphic, which can thus be divided into three types:

1. ***Experience-based behavior sequence:*** In an experience-based behavior sequence a user may interact with a same object (item) multiple times by different behavior types. Different behavior types as well as their orders might indicate user's different intentions, for this type of behavior sequence, a model is expected to capture a user's underlying intentions indicated by different behavior types. The goal here is to predict the next behavior type that the user will exert given an item.

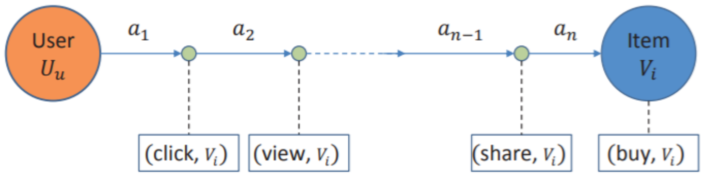


Figure 1.4: Experience-based behavior sequence

2. ***Transaction-based behavior sequence:*** A transaction-based behavior sequence records a series of different behavior objects that a user interacts with, but with a same behavior type. The goal of a sequential recommender system is to recommend the next object that a user will interact with in view of the historical transactions of the user.

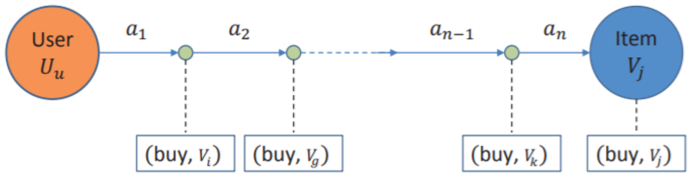


Figure 1.5: Transaction-based behavior sequence

3. ***Interaction-based behavior sequence:*** An interaction-based behavior sequence is a mixture of experience-based and transaction-based behavior sequences It consists of different behavior objects and different behavior types simultaneously. In interaction-based behavioral sequence modeling, a recommender system is expected to understand user preferences more realistically, including different user intents expressed by different behavior types and preferences implied by different behavior objects. Its major goal is to predict the next behavior object that a user will interact with.

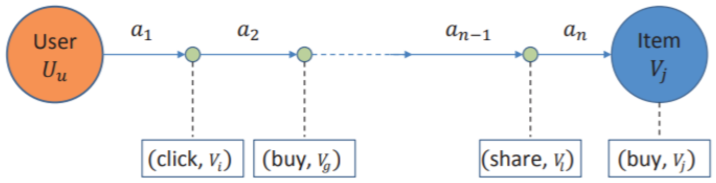


Figure 1.6: Interaction-based behavior sequence

3.4 Approaches

this section discuss a categorization representation of all the approaches for SARS from the technical perspective [23]. This approaches are categorized into eleven atomic classes which are presented in figure 1.7 below.

1. ***Traditional Sequence Models for SARS:*** Traditional sequence models are intuitive solutions to SARS by taking advantage of their natural strength in modeling sequential dependencies among the user-item interactions in a sequence.

   - *Sequential pattern mining:* Sequential pattern-based RS first mine frequent patterns on sequence data and then utilize the mined patterns to guide the subsequent recommendations.

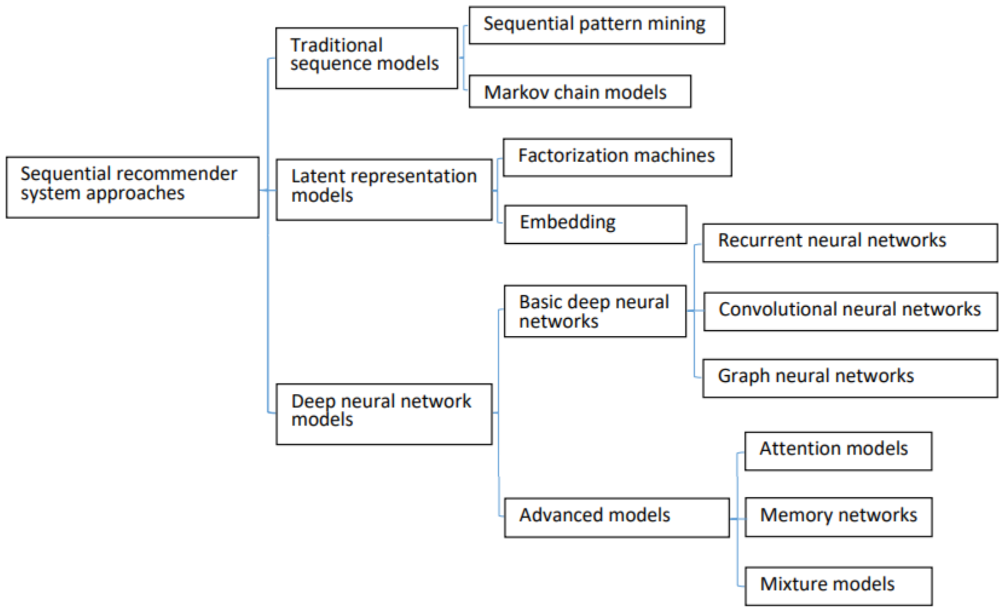

Sequential recommender system approaches
Traditional sequence models
Sequential pattern mining
Markov chain models
Latent representation models
Factorization machines
Embedding
Deep neural network models
Basic deep neural networks
Recurrent neural networks
Convolutional neural networks
Graph neural networks
Advanced models
Attention models
Memory networks
Mixture models

Figure 1.7: Sequential recommender system approaches

- *Markov Chains models:* Markov Chain-based SARS adopt Markov Chains models to model the transitions over user-item interactions in a sequence, for the prediction of the next interaction. According to the specific technique used, Markov Chain-based RS are divided into basic Markov Chain-based approaches and latent Markov embedding-based approaches.

2. ***Latent Representation Models forSARS:*** Latent representation models first learn a latent representation of each user or item, and then predict the subsequent user-item interactions by utilizing the learned representations. As a result, more implicit and complex dependencies are captured in a latent space.

   - *Factorization machines-based SARS:* Usually utilize the matrix or tensor factorization to factorize the observed user-item interactions into latent factors of users and items for recommendations. Such a model is easily affected by the sparsity of the observed data.
   - *Embedding-based SRSs:* Learn a latent representations for each user and item for the subsequent recommendations by encoding all the user-item interactions in a sequence into a latent space. Specifically, some works take the learned latent representations as the input of a network to further calculate an interaction score between users and items, or successive user's actions.

3. ***Deep Neural Network Models for SARS:*** Deep neural networks have natural strength to model and capture the comprehensive relations over different entities (e.g.,users, items, interactions) in a sequence, and thus they nearly dominate SARS in the past few years.

    - *Basic Deep Neural Networks:* The most commonly used deep neural networks for SARS are Recurrent Neural Networks (RNN) due to their natural strength in sequence modelling. Recently, Convolutional Neural Networks (CNN) and Graph Neural Networks (GNN) have also been applied in SARS.

        (a) *RNN-based SARS:* Given a sequence of historical user-item interactions, an RNN-based SARS tries to predict the next possible interaction by modeling the sequential dependencies over the given interactions.

        (b) *CNN-based SARS:* Given a sequence of user-item interactions, a CNN first puts all the embeddings of these interactions into a matrix, and then treats such a matrix as an "image" in the time and latent spaces. Finally, a CNN learns sequential patterns as local features of the image using convolutional filters for the subsequent recommendations.

        (c) *GNN-based SARS:* Captures the complex transitions over user-item interactions in a sequence. Typically a directed graph is first built on the sequence data by taking each interaction as a node in the graph while each sequence is mapped to a path. Then, the embeddings of users or items are learned on the graph to embed more complex relations over the whole graph.

    - *Advanced models:* To address the limitations of SARS built on basic neural network structures, some advanced models are usually combined together with a certain kind of basic deep neural networks to build more powerful SARS which are able to address particular challenges.

        – *Attention model:* Attention models are commonly employed in SARS to emphasize those really relevant and important interactions in a sequence while downplaying those ones irrelevant to the next interaction. They are widely incorporated into shallow networks and RNN to handle interaction sequences with noise.

        – *Memory networks:* Memory networks are introduced into SARS to capture the dependencies between any historical user-item interaction and the next one directly by incorporating an external memory matrix. Such matrix enables it possible to store and update the historical interactions in a sequence more explicitly and dynamically to improve the expressiveness of the model and reduce the interference of those irrelevant interactions.

        – *Mixture model:* A mixture model-based SARS combines different models that excel at capturing different kinds of dependencies to enhance the capability of the whole model in capturing various dependencies for better recommendations.

4 Conclusion

Each day internet users generate huge amounts of data, they use services provided to them through websites and platforms that cater to the needs of each individual specifically. Users don't have the time or power to go through all those data and extract the information they seek neither the companies can, by the use of RS combining the data and the feedback from users better results can be reached satisfying the users or clients and decreasing the needed amount of time and resources.

Computer scientists researched this path over the past decades, many types of RS are attainable to all users and companies around the world, but each has a field that is more compatible with. Based on the data and users and the service provided in general and more importantly the challenges and goals, which explains why SARS is a good fit for platforms that recommend and provide items such as songs, movies, clothes...Etc. That depends on previous choices made by the users themselves. These sequences can be characterized as a sequence of items.

Chapter 2

Hidden Markov Models

1 Introduction

The Hidden Markov Model (HMM) is one of the pillars of statistical modeling of discrete time series, with major successes in applications including speech recognition, computational biology, computer vision, control theory and econometrics, among other disciplines.

The features of HMM are characterized by their simplicity, general mathematical tractability and specifically that the likelihood is simple to compute. The purposes of this chapter is to provide a brief and informal introduction to HMM.

2 Artificial Intelligence

The roots of modern Artificial Intelligence (AI) can be traced back to the classical philosophers of Greece, and their efforts to model human thinking as a system of symbols. In the 1940s, a school of thought called *"Connectionism"* was developed to study the process of thinking. In 1950, *Alan Turing* wrote a paper suggesting how to test a *"thinking machine"*[1]. He believed if a machine could carry on a conversation by way of a teleprinter, imitating a human with no noticeable differences. His paper was followed in 1952 by the *Hodgkin-Huxley* model of the brain as neurons forming an electrical network, with individual neurons firing in all-or-nothing (on/off) pulses. These events, at a conference sponsored by *Dartmouth College* in 1956, helped to spark the concept of Artificial Intelligence.

However, Artificial Intelligence (AI) is the most in demand field in computer science which deals with the simulation of intelligent behavior in computer. AI techniques reside in background as features which improves the overall performance of the system. It can be used along with APIs for software and interfaces for users [28]. It maps between model inputs and

[1] https://www.dataversity.net/brief-history-artificial-intelligence

the parallel outputs for available data using Machine learning by delivering the model inputs and outputs examples repeatedly.

In simplest term AI is manufactured thinking. Intelligence can be viewed as an individual property or quality that can be distinguished from all other properties of an individual [28]. AI can also be noticed in the actions or the ability to perform certain tasks.

3 Machine Learning

The first Machine learning (ML) application that really became mainstream, the spam filter in the 1990s [26]. It's not a self-aware Skynet, but its qualify as Machine learning, It has learned so well that the user rarely needs to flag an email as spam anymore. It was followed by hundreds of ML applications that now quietly power hundreds of products and features, from better recommendations to voice search. Machine learning is the science and art of programming computers so they can learn from data [26].
Arthur Samuel in 1959, defined Machine learning in a general way :

"Machine learning is the field of study that gives computers the ability to learn without being explicitly programmed."

And Tom Mitchell in 1997 in an engineering-oriented one:

"A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E."

3.1 Developing a Machine Learning Model

Machine learning aids in the development of programs that improve their performance for a given task through experience and training. The process of developing ML algorithms may be decomposed into the following steps [27]:

1. ***Data Collecting:*** Select the subset of all available data attributes that might be useful in solving the problem.

2. ***Data preprocessing:*** Present the data in a manner that is understood by the consumer of the data. Preprocessing consists of the following three steps:

   i *Formatting:* The data needs to be presented in a usable format using an industry standard format such as XML, HTML, and SOAP.

   ii *Cleaning:* The data needs to be cleaned by removing, substituting, or fixing corrupt or missing data.

iii *Sampling:* Data need to be sampled by selecting, manipulating and analyzing a representative subset of data without losing information when redundancy is minimized.

3. ***Data Transforming:*** Transform the data specific to the algorithm and the knowledge of the problem. Transformation can be in the form of feature scaling, decomposition, or aggregation.

4. ***Training the algorithm:*** Select the training and testing datasets from the transformed data. An algorithm is trained on the training dataset and evaluated against the test set.

5. ***Testing the algorithm:*** Evaluate the algorithm to test its effectiveness and performance. This step enables quick determination whether any learnable structures can be identified in the data.

6. ***Applying reinforcement learning:*** Most control theoretic applications require a good feedback mechanism for stable operations. In many cases, the feedback data are sparse or unspecific.

7. ***Execute:*** Apply the validated model to perform an actual task of prediction.

3.2 Machine Learning Types

Machine learning systems can be classified according to the amount and type of supervision they get during training [27]. There are four major categories as in figure 2.1

- ***Supervised learning:*** Is a learning mechanism that infers the underlying relationship between the observed data (input data) and a target variable (label) that is subject to prediction. The learning task uses the labeled training data to synthesize the model function that attempts to generalize the underlying relationship between the feature vectors (input) and the supervisory signals (output) [27]. The feature vectors influence the direction and magnitude of change in order to improve the overall performance of the function model. The training data comprise observed input (feature) vectors and a desired output value (figure 2.2) [29]. A well-trained function model based on a supervised learning algorithm can accurately predict the class labels for hidden phenomena embedded in unfamiliar or unobserved data instances. The goal of learning algorithms is to minimize the error for a given set of inputs. However, for a poor-quality training set that is influenced by the accuracy and versatility of the labeled examples, the model may encounter the problem of overfitting, which typically represents poor generalization and erroneous classification.

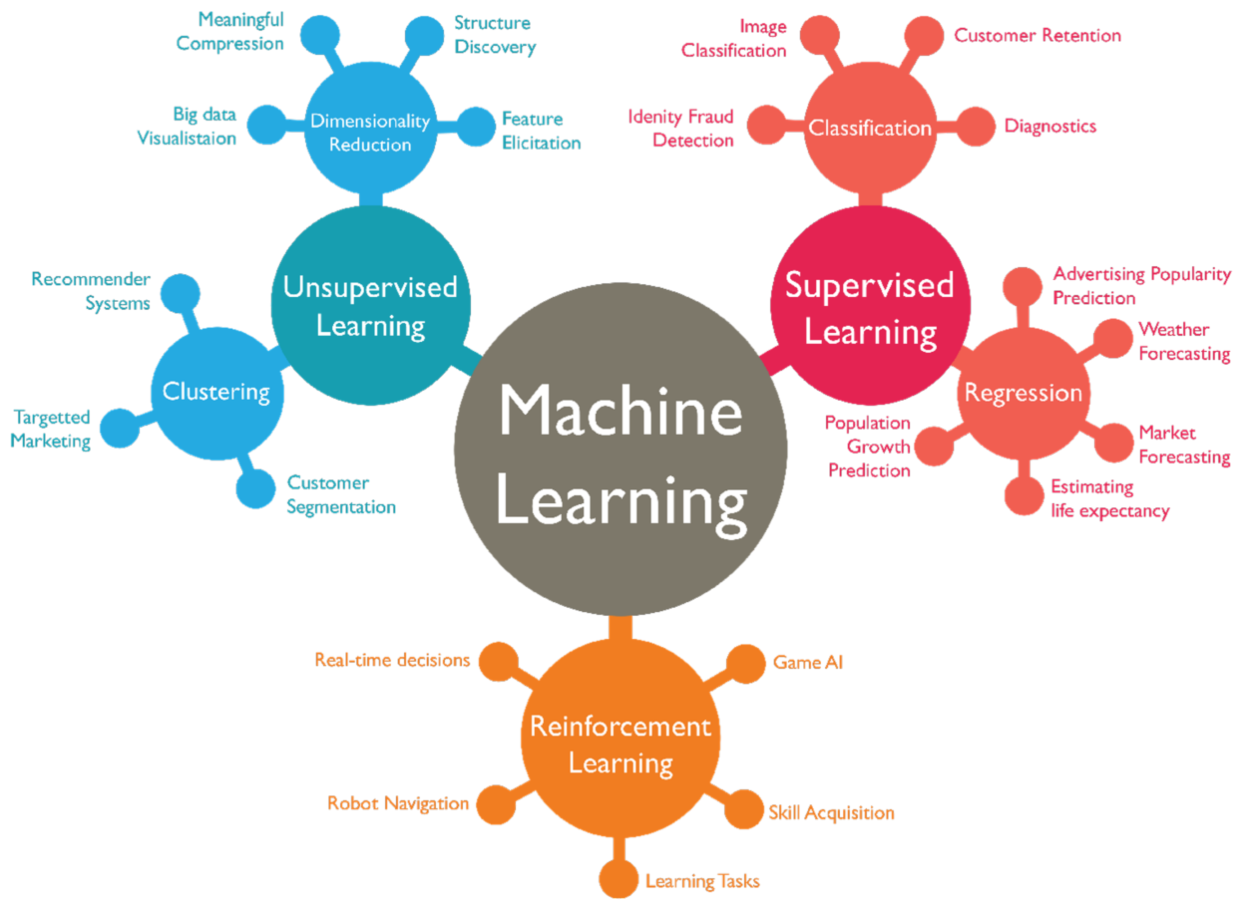


Figure 2.1: Machine Learning Types.

- ***Unsupervised learning:*** Unsupervised learning algorithms are designed to discover hidden structures in unlabeled datasets, in which the desired output is unknown (figure 2.2) [29]. This mechanism has utility value in the areas of data compression, outlier detection, classification, human learning, and so on [27]. The general approach to learning involves training through probabilistic data models. Two popular examples of unsupervised learning are *clustering* and *dimensionality reduction*. In general, an unsupervised learning dataset is composed of inputs but it contains neither target outputs (as in supervised learning) nor rewards from its environment. The goal of ML in this case is to hypothesize representations of the input data for efficient decision making, forecasting, and information filtering and clustering. Unsupervised learning algorithms centered on a probabilistic distribution model generally use maximum likelihood estimation (MLE), maximum a posteriori (MAP), or Bayes methods. Other algorithms that are not based on probability distribution models may employ statistical measurements, quantization error, variance preserving, entropy gaps, and so on.

- ***Semi-supervised learning:*** Uses a combination of a small number of labeled and a large number of unlabeled datasets to generate a model function or classifier. Because the labeling process of acquired data requires intensive skilled human labor inputs, it is expensive and impracticable [27]. In contrast, unlabeled data are relatively inexpensive and readily available. Semi-supervised ML methodology operates somewhere between the guidelines of unsupervised learning (unlabeled training data) and supervised learning (labeled training data) and can produce considerable improvement in learning accuracy. Semi-supervised learning has recently gained greater prominence, owing to the availability of large quantities of unlabeled data for diverse applications to web data, messaging data, stock data, retail data, biological data, images, and so on. This learning methodology can deliver value of practical and theoretical significance, especially in areas related to human learning, such as speech, vision, and handwriting, which involve a small amount of direct instruction and a large amount of unlabeled experience.

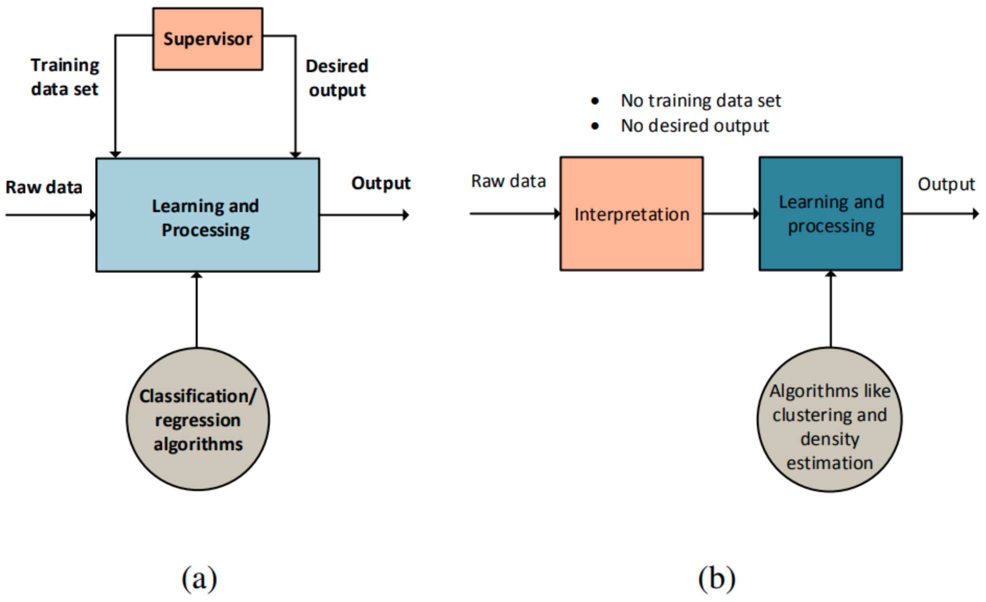


Figure 2.2: Illustrations of the principles of supervised and unsupervised learning.

- ***Reinforcement learning:*** Reinforcement learning methodology involves exploration of an adaptive sequence of actions or behaviors by an intelligent agent in a given environment with a motivation to maximize the cumulative reward [27]. The intelligent agent's action triggers an observable change in the state of the environment. The learning technique synthesizes an adaptation model by training itself for a given set of experimental actions and observed responses to the state of the environment (figure 2.3) [29]. In general, this methodology can be viewed as a control-theoretic trial-and-error learning paradigm with rewards and punishments associated with a sequence of actions. This agent changes its policy based on the collective experience and consequent rewards. reinforcement learning seeks past actions it explored that resulted in rewards. To build an exhaustive database or model of all the possible action reward projections, many unproven actions need to be tried. These untested actions may have to be attempted multiple times before ascertaining their strength. Therefore, striking a balance between exploration of new possible actions and likelihood of failure resulting from those actions is essential.

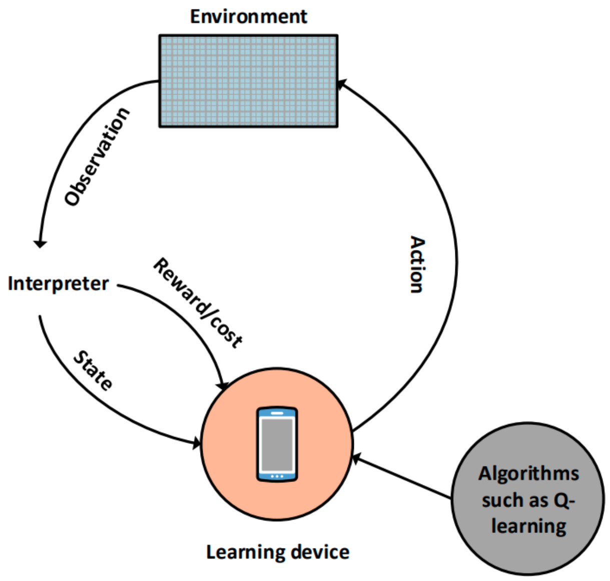


Figure 2.3: Illustration of the principles of a reinforcement learning technique.

3.3 Machine Learning Challenges

Despite living in the prime years of Artificial Intelligence, there are still a lot of obstacles and challenges that will have to be overcome when developing a Machine learning model there are some of the challenges:

- ***Insufficient quantity of training data:*** Machine learning is not quite genius, it takes a lot of data for most Machine learning algorithms to work properly [26]. Even for very simple problems it needs thousands of examples, and for complex problems such as image or speech recognition it might need millions of examples.

- ***The unreasonable effectiveness of data:*** The idea that data matters more than algorithms for complex problems was further popularized by *Peter Norvig et al.* in *"The Unreasonable Effectiveness of Data 2009"* [26]. It should be noted, however, that small and medium-sized datasets are still very common, and it is not always easy or cheap to get extra training data.

- ***Nonrepresentative Training Data:*** The training data should be representative of the new cases to generalize well [26] i.e., the data used for training should cover all the cases that occurred and that is going to occur. By using a non-representative training set, the trained model is not likely to make accurate predictions.

- ***Poor-Quality Data:*** If the training data is full of errors, outliers and noise (e.g., due to poor-quality measurements) [26], it will make it harder for the system to detect the underlying patterns, so the system is less likely to perform well. It is often well worth the effort to spend time cleaning training data.

- ***Irrelevant Features:*** If the training data contains a large number of irrelevant features and enough relevant features, the Machine learning system will not give the results as expected [26]. A critical part of the success of a ML project is coming up with a good set of features to train on. This process, called *feature engineering*, involves the following steps:

  - *Feature selection:* Selecting the most useful features to train on among existing features).
  - *Feature extraction:* Combining existing features to produce a more useful one.
  - Creating new features by gathering new data.

- ***Overfitting the Training Data:*** It means the model is performing well, making likely predictions on the training dataset, but it is not generalized well [26]. overfitting occurs when the model is excessively unpredictable when compared to the noisiness of the training dataset. It can be avoided with the following process:

  - Gathering more training data.
  - Selecting a model with fewer features, a higher degree polynomial model is not preferred compared to the linear model.
  - Fix data errors, remove the outliers, and reduce the number of instances in the training set.

- ***Underfitting the Training Data:*** Underfitting which is opposite to overfitting generally occurs when the model is too simple to understand the base structure of the data [26]. It generally happens when there is less information to construct an exact model and when attempting to build or developing a linear model with non-linear information. Main options to reduce underfitting are:

- Feature Engineering: Feeding better features to the learning algorithm.
- Removing noise from the data.
- Increasing parameters and selecting a powerful model.

4 System models

"Being able to guess" is a concept that is conveyed by the *Greek* word *tokhastikos* from which the word *stochastic* has born. The behavior of any process can invariably be divided into two components: One that follows exactly the model (mental, mathematical,...) that describes its action and other which is unpredictable, and fails from following it. The first component is the *deterministic* part of the event and the second is the *stochastic* one. In case of total absence of determinism, the process is said to be random and, by definition, its outcome cannot be predicted by means of any past information [36].

In practice, it is impossible to have a model that is able to capture all the details of an undergoing process. Deviations between the model output and effective observations will always be present.

Often, dynamical processes in continuous-time, are modeled using systems of differential equations which can lead to an approach designated by state-space model. The usual state-space model formulation regards the deterministic component of a process. The stochastic component can be included by adding a random component whose statistical properties are known or estimated. A continuous-time system is one that has an infinite number of states defined continuously along the independent variable. Notice that, it is also possible to describe discrete-time systems using the same approach [36]. For example, digital systems are particular cases of discrete-time systems with finite number of states. In abstract, the behavior of all automation systems can be described by finite state machines (FSM) which is a mathematical model of computation. It is an abstract machine that can be in exactly one of a finite number of states at any given time [36]. The graphical way to represent the system behavior is designated by state diagram which is represented using circles and arrows. Circles are used to represent the states that the system can exhibit, and the arrows, the way the system navigates through these states.

In some special cases, namely if the inputs to the FSM are chosen randomly with fixed probabilities (or with probabilities that depend only on the current state of the FSM) [35], one can model the behavior of the FSM as a Markov Chains (As shown in figure 2.4 the labels of the arrows in FSM graph are the names of the actions from q_i to q_j and are probabilities in MC graph).

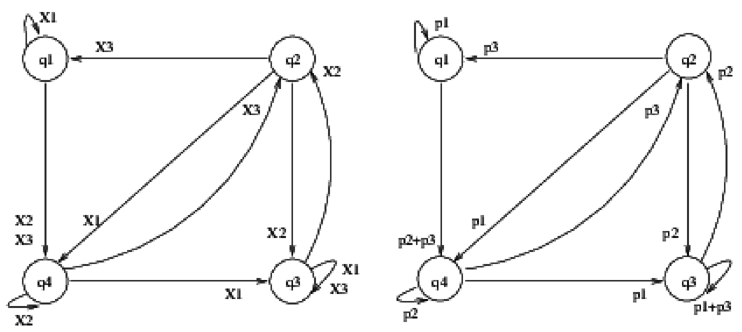


Figure 2.4: Finite state machine and its corresponding Markov Chain

5 Markov chains

Andrey Andreyevich Markov (June 14, 1856 – July 20, 1922) was a Russian mathematician. He is best known for his work on the theory of stochastic Markov processes. His research area later became known as *Markov process* and *Markov Chains. Andrey Andreyevich Markov* introduced the *Markov Chains* in 1906 when he produced the first theoretical results for stochastic processes by using the term *"chain"* for the first time [32].

The HMM is based on augmenting the Markov Chains. A Markov Chains is a model that reveals a meaning about the probabilities of sequences of random variables, states, each of which can take on values from some set [33]. These sets can be words, or tags, or symbols representing anything, like the weather. A Markov chain makes a very strong assumption that predicting the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state. Consider a sequence of state variables $q_1, q_2, q_3, ..., q_i$. A Markov model mimics the Markov assumption on the probabilities of this sequence: That when predicting the future, the past does not matter, only the present.

$$\textbf{Markov Assumption:} \qquad P(q_i = a|q_1, ..., q_{i-1}) = P(q_i = a|q_{i-1}) \tag{2.1}$$

Figure 2.5 shows a Markov chain for assigning a probability $a_{11}, a_{12}, ..., a_{33}$ to a sequence of stats S_1, S_2, S_3. The states are represented as nodes in the graph, and the transitions as edges. The transitions are probabilities: The values of arcs leaving a given state must sum to 1. Given a MC model allows to calculate the probability to any given sequence of states.

Formally, a Markov chain is specified by the following components:

- A set of N *states* $Q = q_1, q_2, q_3, ..., q_N$.
- A *transition probability matrix* A, where $A = a_{11}, a_{12}, ..., a_{nn}$, each a_{ij} representing the probability of moving from state i to state j, so that $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$.

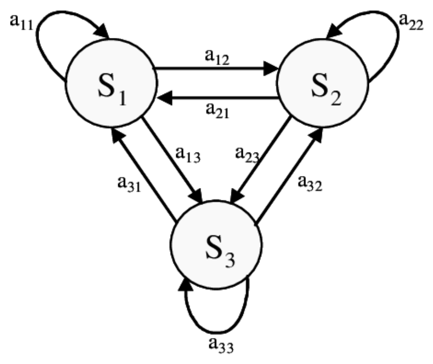


Figure 2.5: An example of Markov chain with 3 states

- An *initial probability distribution* over states $\pi = \pi_1, \pi_2, ..., \pi_N$. π_i is the probability that the Markov Chains will start in state i. Some states may have $\pi = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$.

6 Hidden Markov Model

A Hidden Markov Model (HMM) is a Markov process split into an observable component and an unobserved or latent component. From a statistical standpoint, the use of latent states makes the HMM generic enough to model a variety of complex real-world time series, while the Markovian structure enables relatively simple computational procedures [30].

A Hidden Markov Model consists of two stochastic processes. The first stochastic process is a Markov Chains that is characterized by states and transition probabilities. The states of the chain are externally not visible, therefore *"hidden"*. The second stochastic process produces emissions observable at each moment, depending on a state-dependent probability distribution [33]. It is important to notice that the denomination "hidden" while defining a Hidden Markov Model is referred to the states of the Markov Chains, not to the parameters of the model.

Each Hidden Markov Model is defined by *states, state probabilities, transition probabilities, emission probabilities* and *initial probabilities.* In order to define an HMM completely, the following components have to be defined:

- A set of N *states* $Q = q_1, q_2, q_3, ..., q_N$.

- A *transition probability matrix* A, where $A = a_{11}, a_{12}, ..., a_{nn}$, each a_{ij} representing the probability of moving from state i to state j, so that $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$.

- An *initial probability distribution* over states $\pi = \pi_1, \pi_2, ..., \pi_N$. π_i is the probability that the Markov Chains will start in state i. Some states may have $\pi = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$.

- A sequence of T *observations* $O = o_1, o_2, o_3, ..., o_T$, each one drawn from a vocabulary $V = v_1, v_2, v_3, ..., v_V$.
- A sequence of *observation likelihoods*, also called *emission probabilities* $B = b_{11}, b_{12}, ..., b_{nn}$, each expressing the probability of an observation o_t being generated from a state i $B = b_i(o_t)$.

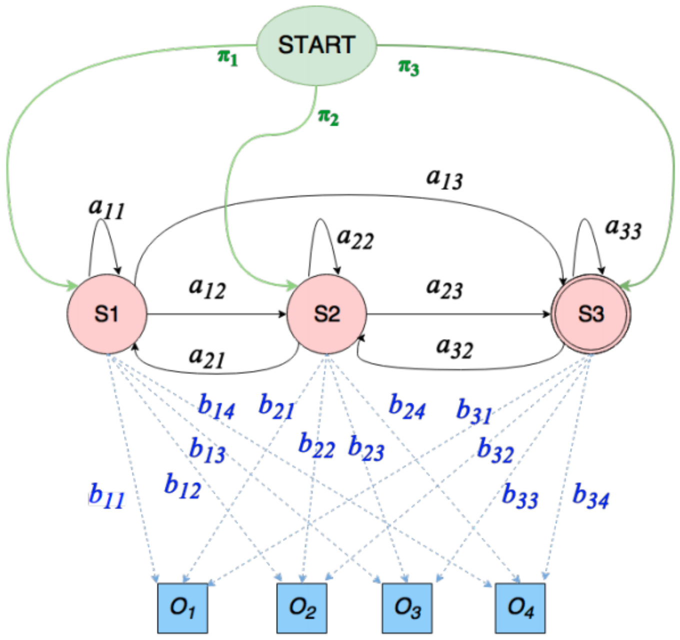


Figure 2.6: HMM graphical model

A first-order Hidden Markov Model instantiates two simplifying assumptions. First, as with a first-order Markov chain, the probability of a particular state depends only on the previous state:

$$\textbf{Markov Assumption:} \quad P(q_i = a|q_1, ..., q_{i-1}) = P(q_i = a|q_{i-1}) \tag{2.2}$$

Second, the probability of an output observation o_i depends only on the state that produced the observation q_i and not on any other states or any other observations:

$$\textbf{Output Independence:} \quad P(o_i = a|q_1, q_i, ..., q_T, o_1, o_i, ..., o_T) = P(o_i|q_i) \tag{2.3}$$

6.1 The Problems of HMM

This section identifies the three fundamental problems [31] that can be solved using HMMs.

- ***Problem 1. Evaluation:*** Given the observation sequence $X = x_1, x_2, x_3, ..., x_t$ and an HMM model $\lambda = (P, B, \pi)$ [31], how do we compute the probability of X? The solution to this problem allows us to select the competing model that best matches the observation sequence.
- ***Problem 2. Decoding:*** Given the observation sequence $X = x_1, x_2, x_3, ..., x_t$ and an HMM model $\lambda = (P, B, \pi)$, how do we find the state sequence $Q = q_1, q_2, q_3, ..., q_t$ that best explains the observations [31]? The solution to this problem attempts to uncover the hidden part of the stochastic model.
- ***Problem 3. Learning:*** How do we adjust the model parameters $\lambda = (P, B, \pi)$ to maximize $P(X|\lambda)$ [31]? The solution to this problem attempts to optimize the model parameters to best describe the observation sequence.

6.2 Solutions to the Three Basic Problems of HMM

6.2.1 Solution to Problem 1

The solution to Problem 1 involves evaluating the probability of observation sequence $X = x_1, x_2, x_3, ..., x_t$ given the model λ that is $P(X|\lambda)$. Consider a state sequence $Q = q_1, q_2, q_3, ..., q_t$, where q_1 and q_t are initial and final states, respectively [31]. The probability of an observation X sequence for a state sequence Q and a model λ can be represented as:

$$P(X|Q, \lambda) = \prod_{i=1}^{n} P(x_t|q_t, \lambda) = b_{x1}(q_1).b_{x2}(q_2).b_{x3}(q_3)...b_{xn}(q_n). \tag{2.4}$$

From the property of Markov chain, the probability of the state sequence can be represented as:

$$P(Q|\lambda) = \pi_{q1}.p_{q1,q2}.p_{21,q3}.p_{q3,q4}...p_{qn-1,qn}. \tag{2.5}$$

Summation over all possible state sequences is as follows:

$$P(X|\lambda) = \sum_{Q} P(X|Q, \lambda) = P(X|Q, \lambda).P(Q|\lambda). \tag{2.6}$$

$$P(X|\lambda) = \sum_{Q} \pi_{q1}.b_{x1}(q_1).p_{q1,q2}.b_{x2}(q_2).p_{q2,q3}...b_{xn}(q_n).p_{qn-1,qn}. \tag{2.7}$$

Unfortunately, direct computation is not very practical, because it requires $2nN^n$ multiplications. At every $t = 1, 2, 3, ..., n, N$ possible states can be reached, which is a huge number.

For example, at $n = 100$ (number of observation sequences) and $N = 5$ (states), there can be $2 \times 100 \times 5^{100} = 10^{72}$ possible computations. Fortunately, an efficient approach, called *the forward algorithm*, achieves the same result.

- ***Forward Algorithm***
  Consider a forward variable $\alpha_t(i)$ that represents the probability of a partial observation sequence up to time t, such that the underlying Markov process is in state S_i at time t, given the HMM model λ:

$$\alpha_t(i) = P(x_1, x_2, x_3, .., x_t, q_t = S_i|\lambda). \tag{2.8}$$

  $\alpha_t(i)$ computed recursively via the following steps:

  1. Initialize the forward probability as a joint probability of state S_i and initial observation x_1. Let $\alpha_1(i) = \pi_i b_i(x_1) \quad for \quad 1 \leqslant i \leqslant N$.
  2. Compute $\alpha_n(j)$ for all states j and $t = n$, using the induction procedure, substituting $t = 1, 2, 3, ..., n,$:

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i).p_{ij}\right] b_j(x_t), 1 \leqslant t \leqslant (n-1), 1 \leqslant j \leqslant N. \tag{2.9}$$

  3. Using the results from the preceding step, compute $P(X|\lambda) = \sum_{j=1}^{N} \alpha_n(j)$.

  The total number of computations involved in evaluating the forward probability is $N^2 n$ rather than $2nN^n$, as required by direct computation. For $n = 100$ and $N = 5$ the total number of computations is $2,500$, which is 10^{69} times smaller in magnitude.

- ***Backward Algorithm***
  For the backward algorithm a backward variable $\beta_t(i)$ can be defined, that represents the probability of a partial observation sequence from time $t+1$ to the end (instead of up to t. as in the forward algorithm), where the Markov process is in state S_i at time t for a given model λ. Mathematically, the backward variable represented as:

$$\beta_t(i) = P(x_{t+1}, x_{t+2}, ..., x_n|q_t = S_i, \lambda). \tag{2.10}$$

  $\alpha_t(i)$ computed recursively via the following steps:

  1. Define $\beta_n(i) = 1 \quad for \quad 1 \leqslant i \leqslant N$.
  2. Compute $\beta_t(i) = \sum_{j=1}^{N} p_{ij} b_j(x_{t+1}) \beta_{t+1}(j)$.

- ***Scaling***
  A practical impediment in modeling long sequences of HMM is the numerical scaling

of conditional probabilities. Efficient computation of conditional probabilities helps in estimating the most likely sequence of states for a given model. For a sufficiently large sequence the probability of observing a long sequence tends to be so extremely small that numerical instability occurs. In most cases, the resulting computations exceed the precision range of essentially any machine (including double-precision). The most common approach for mitigating this situation is to rescale the conditional probabilities, using efficient scaling mechanisms.

6.2.2 Solution to Problem 2

Unlike the solution of Problem 1, identifying the optimal state sequence is a complex problem, because there can be many criteria. Part of the complexity originates from the definition of the measure of optimality, in which several unique criteria are possible. One solution is to identify the states q_t that are most likely to occur individually at time t. This solution attempts to maximize the expected number of correct individual states [31]. To implement the solution to Problem 2, the variable $\gamma_t(i)$ as the probability of being in state S_i at time t, given the observation sequence X and model λ, such that

$$\gamma_t(i) = P(q_t = S_i|X, \lambda) \tag{2.11}$$

Using the definition of conditional probability:

$$\gamma_t(i) = \frac{P(X, q_t = S_i|\lambda)}{P(X, \lambda)} = \frac{P(X, q_t = S_i|\lambda)}{\sum_i^n P(X, q_t = S_i|\lambda)} \tag{2.12}$$

Using the forward-backward variable:

$$\gamma_t(i) = \frac{\alpha_t(i).\beta_t(i)}{\sum_i^N \alpha_t(i).\beta_t(i)} \tag{2.13}$$

Where $\alpha_t(i)$ defines the probability of partial observation $x_1, x_2, x_3, ..., x_t$ and state S_i at time t, and $\beta_t(i)$ defines the remainder of the probability of observation $x_{t+1}, x_{t+2}, x_{t+2}...x_{t+n}$, and state S_i at time t. Using $\gamma_t(i)$,for the individually most likely state q_t^* at each time t solved by calculating the highest probability of being in state S_i at time t, as expressed by the following equation:

$$q_t^* = \arg\max_{1 \leqslant i \leqslant N}[\gamma_t(i)] \quad for \quad \forall t = 1...n \tag{2.14}$$

Although this equation maximizes the expected number of correct states by choosing the most likely state at each time interval, the state sequence itself may not be valid. For instance, in the case of the individually most likely states in the sequence $q_t = S_i$ and $q_{t+1} = S_j$, the transition probability p_{ij} may be 0 and hence not valid. This solution identifies the individually most likely state at any time t without giving any consideration as to the probability of the occurrence of the sequence of states.

One way to address this issue is to maximize the occurrence of a sequence of more than one state. This allows automatic evaluation of valid occurrences of states, while evaluating for the most likely sequence. One widely used scheme is to find the single most likely sequence of states that ultimately results in maximizing $P(X, Q|\lambda)$. This technique, which is based on dynamic programming, is called a *Viterbi algorithm.* To find the single best state sequence, define a variable $\delta_t(i)$ that represents the highest probability along one state sequence (path) that accounts for first t observations and that ends in state S_i, as follows:

$$\delta_t(i) = \max_{q_1,q_2,\ldots,q_{t-1}} (q_1, q_2, ..., q_t = S_i, x_1, x_2, ..., x_t|\lambda). \tag{2.15}$$

$\delta_{t+1}(i)$ can be calculated by induction:

$$\delta_{t+1}(i) = \max_i \left[\delta_t(i).p_{ij}\right].b_j(x_{t+1}). \tag{2.16}$$

from which it is clear that to retrieve the state sequence, it must to track the state that maximizes $\delta_i(i)$ at each time t. This is done by constructing an array $\psi_{t+1}(j)$ that defines the state at time t from which a transition to state S_j maximizes the probability $\delta_{t+1}(j)$. Mathematically, this can be represented as:

$$\psi_{t+1}(i) = \arg\max_{1\leqslant i\leqslant N} \left[\delta_t(i).p_{ij}\right]. \tag{2.17}$$

The complete procedure for finding the best state sequence consists of the following steps:

- ***Initialization***

$$\delta_1(i) = \pi_i.b_i(x_1), 1 \leqslant I \leqslant N \tag{2.18}$$

$$\psi_1(i) = 0 \tag{2.19}$$

- ***Recursion***

$$\delta_1(i) = \max_{1\leqslant i\leqslant N} \left[\delta_{t-1}.p_{ij}\right].b_j(x_t); 1 \leqslant j \leqslant N; 2 \leqslant t \leqslant n \tag{2.20}$$

$$\psi_t(j) = \arg\max_{1\leqslant i\leqslant N} \left[\delta_{t-1}(i).p_{ij}\right] 1 \leqslant j \leqslant N; 2 \leqslant t \leqslant n \tag{2.21}$$

- ***Termination***

$$p^* = \max_{1\leqslant i\leqslant N} [\delta_n(i)] \tag{2.22}$$

$$q_n^* = \arg\max_{1\leqslant i\leqslant N} [\delta_n(i)] \tag{2.23}$$

- ***State Sequence Backtracking***

$$q_t^* = \psi_{t+1}(q_{t+1}^*); t = n-1, n-2, n-3, ..., 1. \tag{2.24}$$

The *Viterbi algorithm* is similar to the *forward procedure*, except that it uses maximization over previous states instead of a summation.

6.2.3 Solution to Problem 3

The solution to Problem 3 involves a method for adjusting the model parameters (P, B, π) to maximize the probability of an observation sequence for a given model. In practice there is no well-known method that maximizes the probability of observation sequence. However, you can select $\lambda = (P, B, \pi)$, such that $P(X|\lambda)$ is locally maximized, using an iterative method, such as the *Baum-Welch algorithm* [31].

To specify the re-estimation of HMM parameters, you define the variable $\gamma_t(i,j)$ as the probability of being in state S_i at time t and in S_j at time $t+1$ for a given model λ and observation sequence X, such that:

$$\gamma_t(i,j) = P(q_t = S_i, q_{t+1} = S_j|X, \lambda) \tag{2.25}$$

Using the definition of the forward-backward algorithm:

$$\gamma_t(i,j) = \frac{\alpha_t(i).p_{ij}.b_j(x_{t+1}).\beta_{t+1}(j)}{P(X|\lambda)} \tag{2.26}$$

$$\gamma_t(i,j) = \frac{\alpha_t(i).p_{ij}.b_j(x_{t+1}).\beta_{t+1}(j)}{\sum_i^N \sum_j^N \alpha_t(i).p_{ij}.b_j(x_{t+1}).\beta_{t+1}(j)} \tag{2.27}$$

As defined by Equation (2.13), $\gamma_t(i)$ is the probability of being in state S_i at time t, given the observation sequence and model. Using this equation, $\gamma_t(i)$ to $\gamma_t(i,j)$ can be related by summing over j as:

$$\gamma_t(i) = \sum_j^N \gamma_t(i,j) \tag{2.28}$$

By summing $\gamma_t(i)$ over time t, you can quantify the number of times state S_i is visited or, alternatively, the expected number of transitions made from state S_i . Similarly, summation of$\gamma_t(i,j)$ over time t reveals the expected number of transitions from state S_i to state S_j . Given $\gamma_t(i)$, $\gamma_t(i,j)$ and the current model λ, you can build the method to reestimate the parameters of the HMM model $(\overline{\lambda})$. The method can be broken down as follows:

1. At time $t = 1$ the expected frequency at state S_i is given by $\overline{\pi_{\mathrm{i}}} = \gamma_1(i) \forall i = (1, 2, 3, ..., N)$.

2. The probability of transiting from state $S_i$to state S_j , which is the desired value of $\overline{\mathrm{p_{ij}}}$, is given by:

$$\overline{\mathrm{p_{ij}}} = \frac{\sum_{t=1}^{n-1} \gamma_t(i,j)}{\sum_{t=1}^{n-1} \gamma_t(i)} \forall i, j = (1, 2, 3, ..., N). \tag{2.29}$$

   The numerator is the re-estimated value of the expected number of transitions from state S_i to state S_j, the denominator is the expected number of transitions from S_i to any state.

3. The probability of observing symbol k, given that the model is in state S_j, is given by

$$\overline{\mathrm{b_j(k)}} = \frac{\sum_{t=1,x_t=k}^{n} \gamma_t(j)}{\sum_{t=1}^{n} \gamma_t(j)} \forall k = (1, 2, 3, ..., M). \tag{2.30}$$

   The numerator of the re-estimated $\overline{\mathrm{b_j(k)}}$ is the expected number of times the model is in state S_j with observation symbol k, the denominator is the expected number of times the model is in state S_j.

With this method, you use the current model $\lambda = (P, B, \pi)$ to re-estimate the new model $\overline{\lambda = (P, B, \pi)}$, as described by the previous three steps. The re-estimation process is an iterative method consisting of the following steps:

1. Initialize $\lambda = (P, B, \pi)$ with a best guess or random value, or use the existing model.
2. Compute $\alpha_t(i), \beta_t(i), \gamma_t(i), \gamma_t(i, j)$
3. Re-estimate the model $\overline{\lambda = (P, B, \pi)}$
4. If $P(X|\overline{\lambda}) > (P(X|\lambda)$, repeat step 2.

The final result of this re-estimation process is called the maximum likelihood estimation (MLE) of the parameters of the HMM. The forward-backward algorithm yields only the local maximum.

6.3 Types of Hidden Markov Models

Different kinds of structures for HMMs can be used. The structure is defined by the transition matrix, A.
The most general structure is the *ergodic* or *fully connected HMM* [34]. In this model every state can be reached from every other state of the model see figure 2.7. The ergodic model has the property $0 < a_{ij} < 1$ where the *"zero"* and the *"one"* have been excluded in order to fulfill the ergodic property. The state transition matrix, A, for an ergodic model, can be described by:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

The second structure is the *left-right model* or *Bakis model* [34], see figure 2.8 *a* and *b* the property for a left-right model is:

$$a_{ij} = 0, j < i.$$

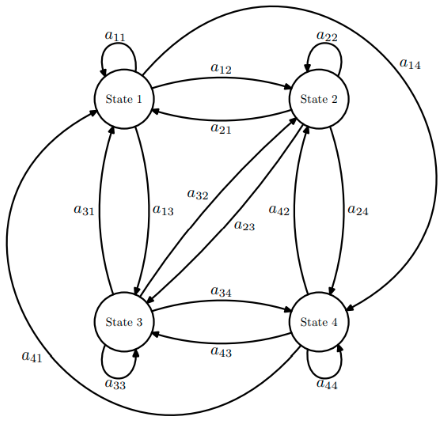


Figure 2.7: ergodic (fully connected HMM)

This implies that no transitions can be made to previous states. The lengths of the transitions are usually restricted to some maximum length $\triangle$.

$$a_{ij} = 0, j > i + \triangle. \tag{2.31}$$

Note that for a left-right model, the state transitions coefficients for the last state have the following property:

$$a_{NN} = 1 \tag{2.32}$$

$$a_{Nj} = 0, \quad j < N \tag{2.33}$$

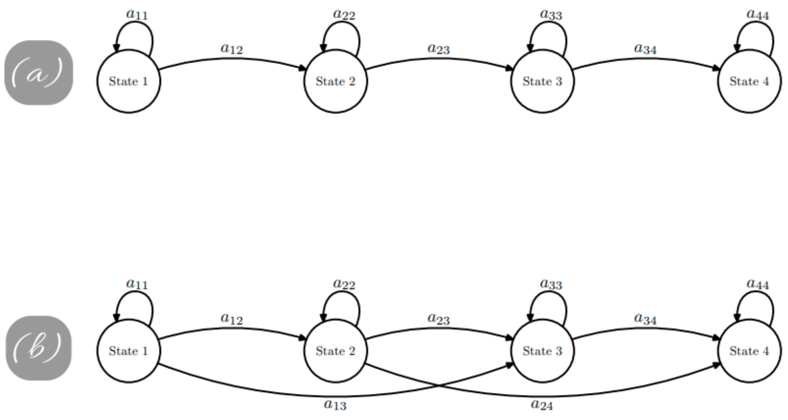


Figure 2.8: left-right model (Bakis model)

7 HMM application example

Using the theory of this chapter as an example we can apply HMM to weather forecast knowing that HMM is the 5-tuple $\triangle = (S, \Phi, A, B, \pi)$. Note that this components are often called parameters of HMM in which $A, B \quad and \quad \pi$ are essential parameters. Going back weather example is the most common example for understanding how HMM work, suppose you need to predict how weather tomorrow is: *sunny, cloudy or rainy* since you know only observations about the humidity: *dry, dryish, damp, soggy.* The HMM is totally determined based on its parameters according to weather example.
We have $S = (S_1$=sunny, S_2=cloudy, S_3=rainy), $\Phi = (\phi_1$=dry, ϕ_2=dryish, ϕ_3=damp, ϕ_4=soggy). The transition probability matrix, A, for this example, can be described by:

$$A = \begin{array}{c} \\ sunny \\ cloudy \\ rainy \end{array} \begin{array}{c} \begin{array}{ccc} sunny & cloudy & rainy \end{array} \\ \begin{pmatrix} a_{11} = 0.50 & a_{12} = 0.25 & a_{13} = 0.25 \\ a_{21} = 0.30 & a_{22} = 0.40 & a_{23} = 0.30 \\ a_{31} = 0.25 & a_{32} = 0.25 & a_{33} = 0.50 \end{pmatrix} \end{array}$$

From the transition probability matrix, A we can see that:

$$a_{11} + a_{12} + a_{13} = 1$$
$$a_{21} + a_{22} + a_{23} = 1$$
$$a_{31} + a_{32} + a_{33} = 1$$

The initial probability distribution specified as uniform distribution represented as:

sunny	cloudy	rainy
$\pi_1 = 0.34$	$\pi_2 = 0.33$	$\pi_3 = 0.33$

Table 2.1: Initial probability distribution

from the table 2.1 we have $\pi_1 + \pi_2 + \pi_3 = 1$
And the emission probability matrix B:

$$B = \begin{array}{c} \\ sunny \\ cloudy \\ rainy \end{array} \begin{array}{c} \begin{array}{cccc} dry & dryish & damp & soggy \end{array} \\ \begin{pmatrix} b_{11} = 0.60 & b_{12} = 0.20 & b_{13} = 0.15 & b_{14} = 0.05 \\ b_{21} = 0.25 & b_{22} = 0.25 & b_{23} = 0.25 & b_{24} = 0.25 \\ b_{31} = 0.o5 & b_{32} = 0.10 & b_{33} = 0.35 & b_{34} = 0.50 \end{pmatrix} \end{array}$$

From the the emission probability matrix B we have:

$$b_{11} + b_{12} + b_{13} + b_{14} = 1$$
$$b_{21} + b_{22} + b_{23} + b_{24} = 1$$
$$b_{31} + b_{32} + b_{33} + b_{34} = 1$$

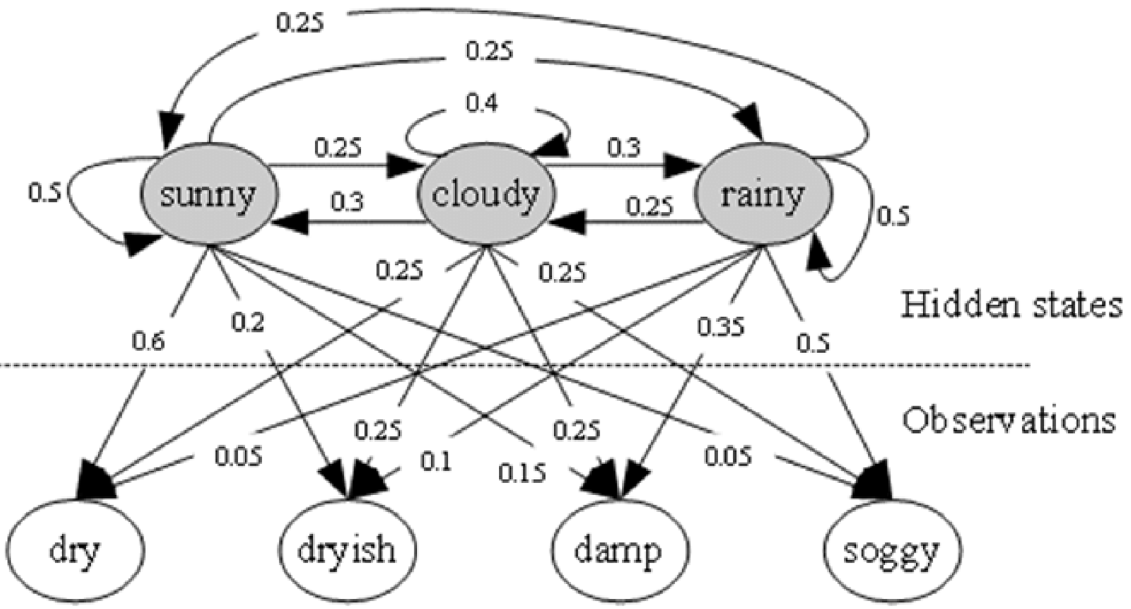


Figure 2.9: HMM for weather forecast

From the above data, we can embody the HMM in a graphical model showed in figure 2.9 With the defined HMM , given a sequence of humidity observation *(soggy, dry, dryish, dry, dry, dryish, damp, soggy, soggy, dry, dryish, damp, soggy)*, It can calculate the *Viterbi* path or the most likely sequence of states *(weather forecast-start, rainy, sunny, sunny, sunny, sunny, sunny, rainy, rainy, rainy, sunny, sunny, rainy, rainy)* and It's probability 23.94287, then basing on it we can determine the next day's weather using the given probabilities $(A, B \quad and \quad \pi)$.

8 Conclusion

Computers are powerful tools to the aid of human beings when used the right way, enabling them to achieve great results with sufficient data and specialized models. Through modeling, scientists can create many different models as solutions to different problems they encounter in every field possible.

Models mainly are a part of AI systems that empower computers and recreate results that humans can do but are laborious, time and resource consuming. ML is an important aspect of it, which helps to recreate the learning process humans can follow.

ML has many models that it can exploit, one of which is HMM a statistical model that is fairly easier to understand and use and has a straightforward approach to problem solving using relations between what is known *"observables"* and the unknown *"Hidden states"* to determine the later one, by calculating the most likely hidden state.

Chapter 3

An HMM Architecture for SARS Development

1 Introduction

Recommender systems can be used to enhance user experience by making personalized and useful recommendations for each user. In order to personalize recommendations, traditional Recommender systems often need to build up a user profile.

However, this may not be always possible: new users may not have any profile, or not logged, or deleted their browsing history. An alternative is to make Sequence-Aware Recommender Systems. In this setting, the recommender system makes recommendations based only on the user sequences. Hidden Markov Model (HMM) were recently proposed for the Sequence-Aware Recommender Systems task that had significant improvements over traditional recommendation models. In this chapter, the application of HMM for SARS are further studied.

2 Sequence-Aware Recommendations with HMMs

This work's focus is the preferences of users, which is presented as time-stamped sequences. These sequences consist of a list of items (Tracks) the user previously had an interest in or interacted with, this model's purpose is to predict and recommend future interests based on the sequences (see figure 3.1).

To incorporate the sequential information of user's preferences into recommendation, Hidden Markov Model models are adopted. Treating a user's preferences sequence as a sentence in a statistical language model, the probability of the current item depends only on the probability of the former item based on Markov assumption (first order Markov chain). To reach our goal which is to predict next-item using the hole sequence we integrated the concept of high-order Markov Chains (the probability of the current item depends on the probability of all items in the sequence).

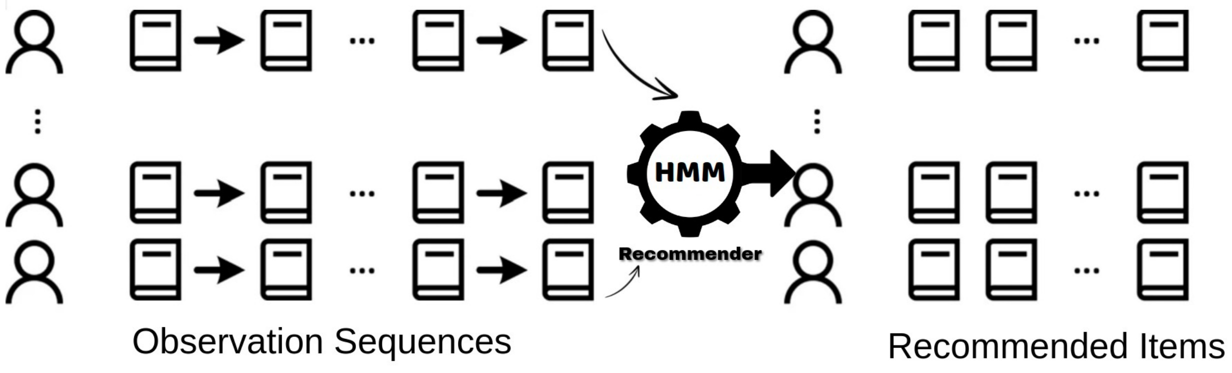


Figure 3.1: Illustration of the studied recommendation system

2.1 HMM Model architecture

This section shows us the the architecture of the proposed Hidden Markov Model. The figure 3.2 bellow illustrates the proposed graphical Hidden Markov Model of the proposed architecture.

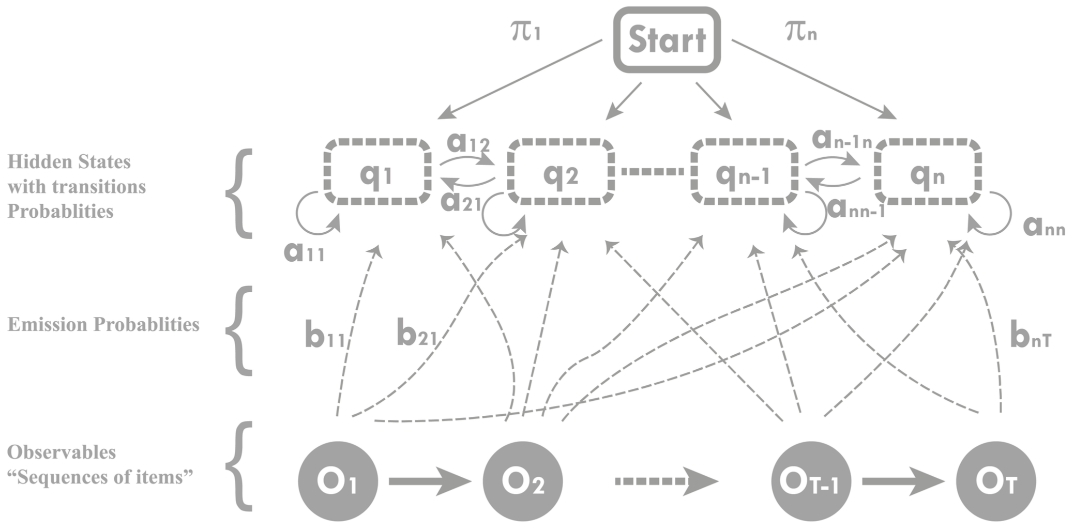


Figure 3.2: HMM graphical model

The table 3.1 bellow enumerates the parameters and their signification and description of the proposed architecture.

Parameter	signification	Description
N	Number of hidden states	Number of hours (12 or 24)
T	Number of observations	Number of unique tracks
π_i	Start probabilities	occurrence of each hidden state
a_{ij}	Transition probabilities	probability of navigating from each state
b_{ij}	Emission probabilities	probability between states and observations
q_i	Hidden states	Timestamped values (hours)
o_i	Observation sequence	user track sequences

Table 3.1: Hidden Markov Model parameters

2.2 DataSet

There are several kinds of data sets for RS. This paper conducted a set of experiments with real usage of *Last.fm*[1] dataset which is collected from Last.fm API[2] by *Oscar Celma*, about user's music listening activity to make personalized recommendations at the online radio station.

- ***Data Format:***
  The data is formatted one entry per line as follows (tab separated):
  (user id, timestamp, artist-id, artist name, track id, track-name)

- ***Data Statistics:***
  Total Lines: 19,150,868
  Unique Users: 992

In this work we only used user-id, timestamped and track-name columns in addition of a reasonable set of users.

2.3 Development process

This section of the chapter explains how we develop Sequence-Aware Recommender Systems using Hidden Markov Model passing by three main algorithms(data processing, parameter initialization and the Next-Item recommendation):

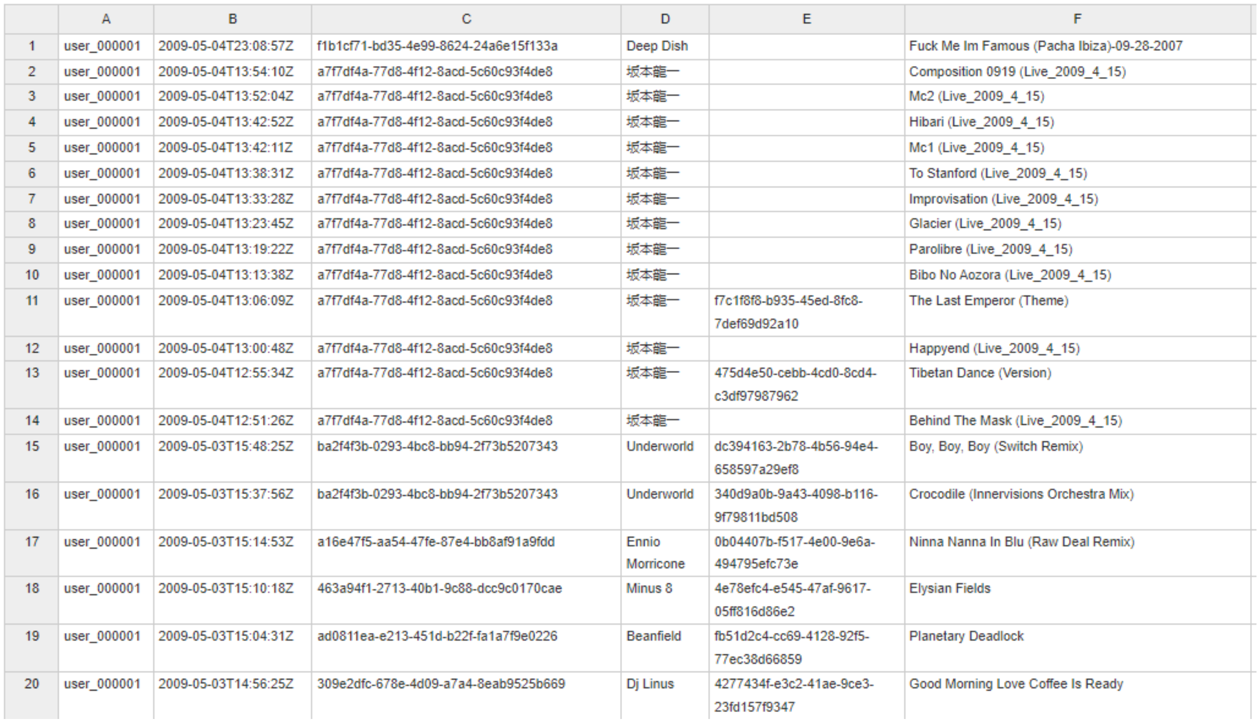
[1]ocelma.net/MusicRecommendationDataset/lastfm-1K.html
[2]last.fm/api

2.3.1 Data processing

Algorithm 1: Data processing

```
1: Split the dataset.
2: Extract the set of observations ($O = o_1, o_2, o_3, ..., o_T$) (unique tracks).
3: Extract observation sequences (user tracks sequences).
4: Extract hidden states ($Q = q_1, q_2, q_3, ..., q_N$)(timestamps).
5: Encode the observation.
```

The first algorithm describes the first step of developing Sequence-Aware Recommender Systems with HMM witch is *data processing*, since we are working with last.fm dataset considering it's format our interest is in the sequences of tracks as observations and the timestamps as hidden states grouped by the number of hours, and due to the huge amount of data in last.fm we have reduced it (reasonable number of users to work with), as shown in figure 3.3 a slice of the last.fm dataset, which in this work only took a reasonable amount of data based on the number of users.



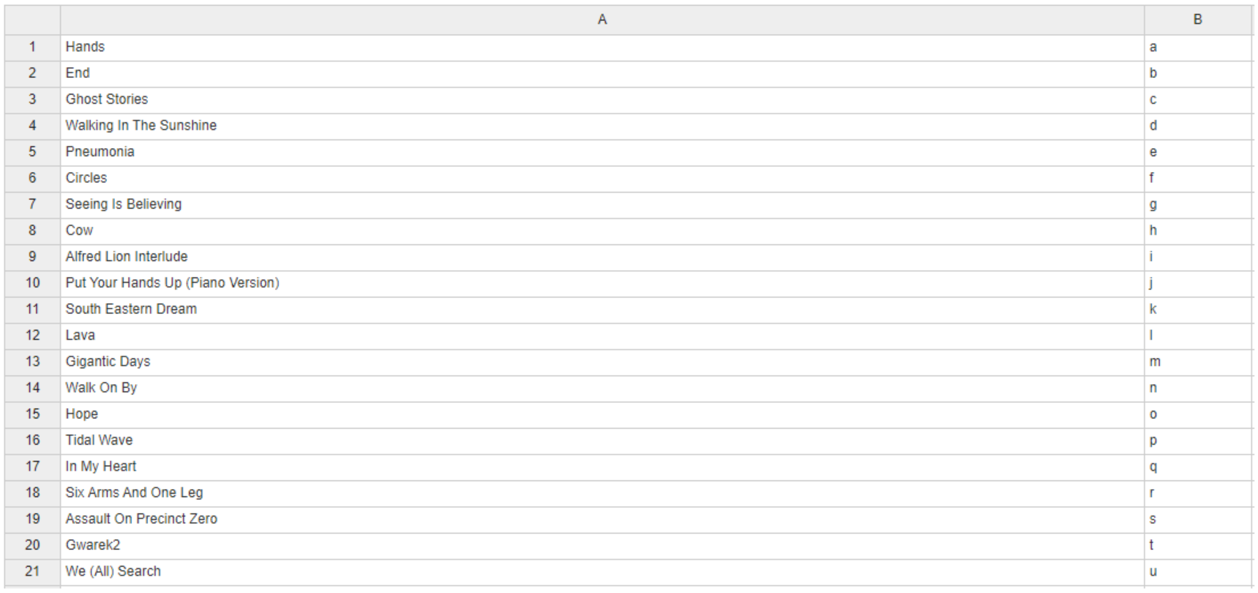
	A	B	C	D	E	F
1	user_000001	2009-05-04T23:08:57Z	f1b1cf71-bd35-4e99-8624-24a6e15f133a	Deep Dish		Fuck Me Im Famous (Pacha Ibiza)-09-28-2007
2	user_000001	2009-05-04T13:54:10Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Composition 0919 (Live_2009_4_15)
3	user_000001	2009-05-04T13:52:04Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Mc2 (Live_2009_4_15)
4	user_000001	2009-05-04T13:42:52Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Hibari (Live_2009_4_15)
5	user_000001	2009-05-04T13:42:11Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Mc1 (Live_2009_4_15)
6	user_000001	2009-05-04T13:38:31Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		To Stanford (Live_2009_4_15)
7	user_000001	2009-05-04T13:33:28Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Improvisation (Live_2009_4_15)
8	user_000001	2009-05-04T13:23:45Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Glacier (Live_2009_4_15)
9	user_000001	2009-05-04T13:19:22Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Parolibre (Live_2009_4_15)
10	user_000001	2009-05-04T13:13:38Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Bibo No Aozora (Live_2009_4_15)
11	user_000001	2009-05-04T13:06:09Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一	f7c1f8f8-b935-45ed-8fc8-7def69d92a10	The Last Emperor (Theme)
12	user_000001	2009-05-04T13:00:48Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Happyend (Live_2009_4_15)
13	user_000001	2009-05-04T12:55:34Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一	475d4e50-cebb-4cd0-8cd4-c3df97987962	Tibetan Dance (Version)
14	user_000001	2009-05-04T12:51:26Z	a7f7df4a-77d8-4f12-8acd-5c60c93f4de8	坂本龍一		Behind The Mask (Live_2009_4_15)
15	user_000001	2009-05-03T15:48:25Z	ba2f4f3b-0293-4bc8-bb94-2f73b5207343	Underworld	dc394163-2b78-4b56-94e4-658597a29ef8	Boy, Boy, Boy (Switch Remix)
16	user_000001	2009-05-03T15:37:56Z	ba2f4f3b-0293-4bc8-bb94-2f73b5207343	Underworld	340d9a0b-9a43-4098-b116-9f79811bd508	Crocodile (Innervisions Orchestra Mix)
17	user_000001	2009-05-03T15:14:53Z	a16e47f5-aa54-47fe-87e4-bb8af91a9fdd	Ennio Morricone	0b04407b-f517-4e00-9e6a-494795efc73e	Ninna Nanna In Blu (Raw Deal Remix)
18	user_000001	2009-05-03T15:10:18Z	463a94f1-2713-40b1-9c88-dcc9c0170cae	Minus 8	4e78efc4-e545-47af-9617-05ff816d86e2	Elysian Fields
19	user_000001	2009-05-03T15:04:31Z	ad0811ea-e213-451d-b22f-fa1a7f9e0226	Beanfield	fb51d2c4-cc69-4128-92f5-77ec38d66859	Planetary Deadlock
20	user_000001	2009-05-03T14:56:25Z	309e2dfc-678e-4d09-a7a4-8eab9525b669	Dj Linus	4277434f-e3c2-41ae-9ce3-23fd157f9347	Good Morning Love Coffee Is Ready

Figure 3.3: The last.fm Data slice

Secondly, we extract the observation (unique tracks) besides the encoding step[3] for

[3]pomegranate.readthedocs.io/en/latest/Distributions.html?highlight=DiscreteDistribution

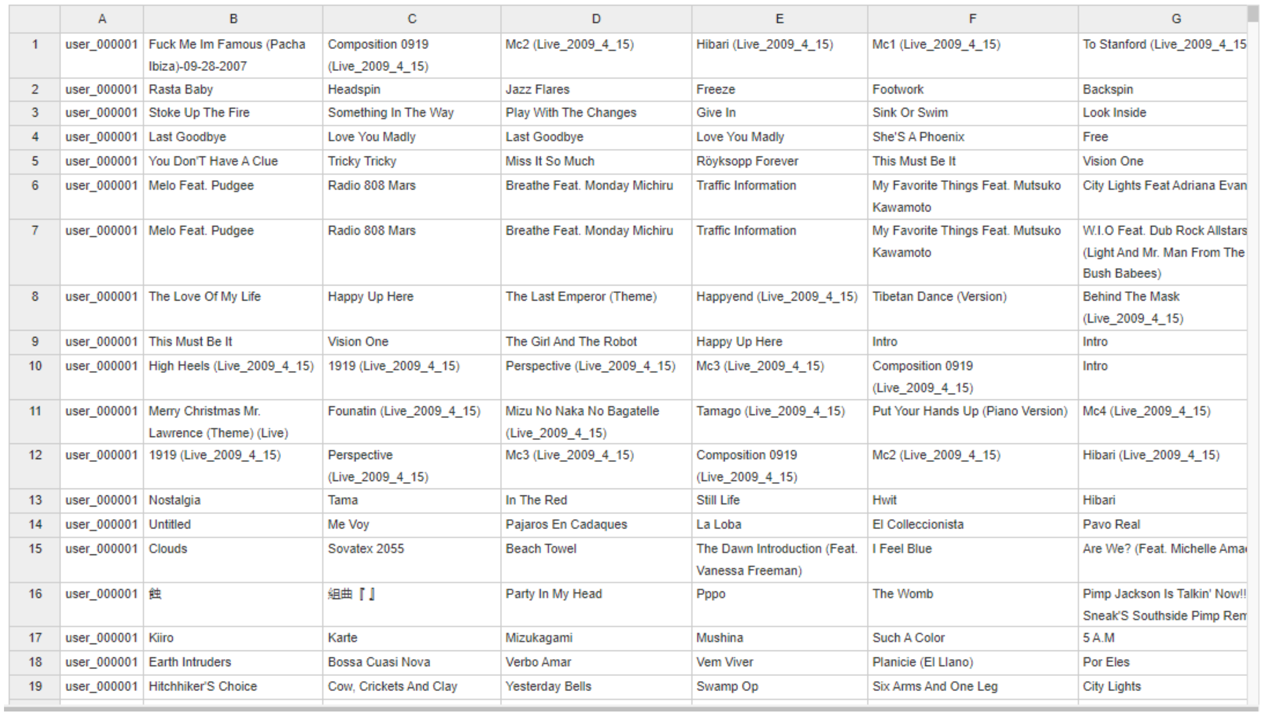
assigning them to the HMM, Then extract both the sequences of observations, and the hidden states in our model we have two set oh hidden states, the 12 hidden state model $hour_1, hour_2, ..., hour_{12}$ and the 24 hidden state model $hour_1, hour_2, ..., hour_{24}$ knowing that in the 12 hidden state each state regroup two hours and one hour in the 24 hidden state model.Next we split the sequence of observation into a user's input sequence and a test sequence that we aimed to predict and evaluate the HMM with. The length of the input sequence is set to 60 and the test sequence is set to 20.
The figure 3.4 shows a set of unique tracks(songs) or refers to as the observations of the proposed model and their encoding allowing the model to properly understand and read the data.



	A	B
1	Hands	a
2	End	b
3	Ghost Stories	c
4	Walking In The Sunshine	d
5	Pneumonia	e
6	Circles	f
7	Seeing Is Believing	g
8	Cow	h
9	Alfred Lion Interlude	i
10	Put Your Hands Up (Piano Version)	j
11	South Eastern Dream	k
12	Lava	l
13	Gigantic Days	m
14	Walk On By	n
15	Hope	o
16	Tidal Wave	p
17	In My Heart	q
18	Six Arms And One Leg	r
19	Assault On Precinct Zero	s
20	Gwarek2	t
21	We (All) Search	u

Figure 3.4: encoded-tracks

Figure 3.5 represent a portion of user-track sequences or the observation sequences, the length of each sequence is set to 80, the choice of this split was random.



	A	B	C	D	E	F	G
1	user_000001	Fuck Me Im Famous (Pacha Ibiza)-09-28-2007	Composition 0919 (Live_2009_4_15)	Mc2 (Live_2009_4_15)	Hibari (Live_2009_4_15)	Mc1 (Live_2009_4_15)	To Stanford (Live_2009_4_15
2	user_000001	Rasta Baby	Headspin	Jazz Flares	Freeze	Footwork	Backspin
3	user_000001	Stoke Up The Fire	Something In The Way	Play With The Changes	Give In	Sink Or Swim	Look Inside
4	user_000001	Last Goodbye	Love You Madly	Last Goodbye	Love You Madly	She'S A Phoenix	Free
5	user_000001	You Don'T Have A Clue	Tricky Tricky	Miss It So Much	Röyksopp Forever	This Must Be It	Vision One
6	user_000001	Melo Feat. Pudgee	Radio 808 Mars	Breathe Feat. Monday Michiru	Traffic Information	My Favorite Things Feat. Mutsuko Kawamoto	City Lights Feat Adriana Evan
7	user_000001	Melo Feat. Pudgee	Radio 808 Mars	Breathe Feat. Monday Michiru	Traffic Information	My Favorite Things Feat. Mutsuko Kawamoto	W.I.O Feat. Dub Rock Allstars (Light And Mr. Man From The Bush Babees)
8	user_000001	The Love Of My Life	Happy Up Here	The Last Emperor (Theme)	Happyend (Live_2009_4_15)	Tibetan Dance (Version)	Behind The Mask (Live_2009_4_15)
9	user_000001	This Must Be It	Vision One	The Girl And The Robot	Happy Up Here	Intro	Intro
10	user_000001	High Heels (Live_2009_4_15)	1919 (Live_2009_4_15)	Perspective (Live_2009_4_15)	Mc3 (Live_2009_4_15)	Composition 0919 (Live_2009_4_15)	Intro
11	user_000001	Merry Christmas Mr. Lawrence (Theme) (Live)	Founatin (Live_2009_4_15)	Mizu No Naka No Bagatelle (Live_2009_4_15)	Tamago (Live_2009_4_15)	Put Your Hands Up (Piano Version)	Mc4 (Live_2009_4_15)
12	user_000001	1919 (Live_2009_4_15)	Perspective (Live_2009_4_15)	Mc3 (Live_2009_4_15)	Composition 0919 (Live_2009_4_15)	Mc2 (Live_2009_4_15)	Hibari (Live_2009_4_15)
13	user_000001	Nostalgia	Tama	In The Red	Still Life	Hwit	Hibari
14	user_000001	Untitled	Me Voy	Pajaros En Cadaques	La Loba	El Colleccionista	Pavo Real
15	user_000001	Clouds	Sovatex 2055	Beach Towel	The Dawn Introduction (Feat. Vanessa Freeman)	I Feel Blue	Are We? (Feat. Michelle Ama
16	user_000001	蝕	組曲『』	Party In My Head	Pppo	The Womb	Pimp Jackson Is Talkin' Now!! Sneak'S Southside Pimp Rem
17	user_000001	Kiiro	Karte	Mizukagami	Mushina	Such A Color	5 A.M
18	user_000001	Earth Intruders	Bossa Cuasi Nova	Verbo Amar	Vem Viver	Planicie (El Llano)	Por Eles
19	user_000001	Hitchhiker'S Choice	Cow, Crickets And Clay	Yesterday Bells	Swamp Op	Six Arms And One Leg	City Lights

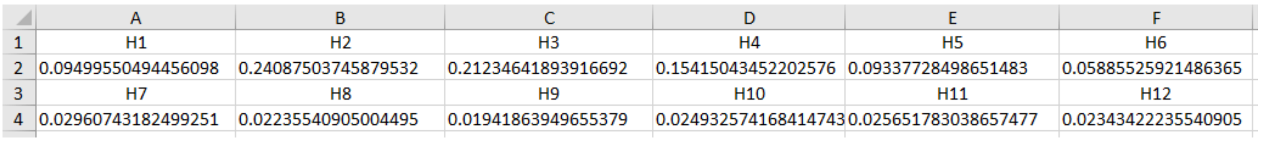
Figure 3.5: User-Track sequences

2.3.2 Parameter initialization

Algorithm 2: Initialize the model parameter

```
1: Compute the start probability $\pi = \pi_1, \pi_2, ..., \pi_N$.
2: Compute transition probability $A = a_{11}, a_{12}, ..., a_{NN}$.
3: Compute emission probability $B = b_{11}, b_{12}, ..., b_{NT}$.
4: Initialize HMM $\lambda = (A, B, \pi)$.
```

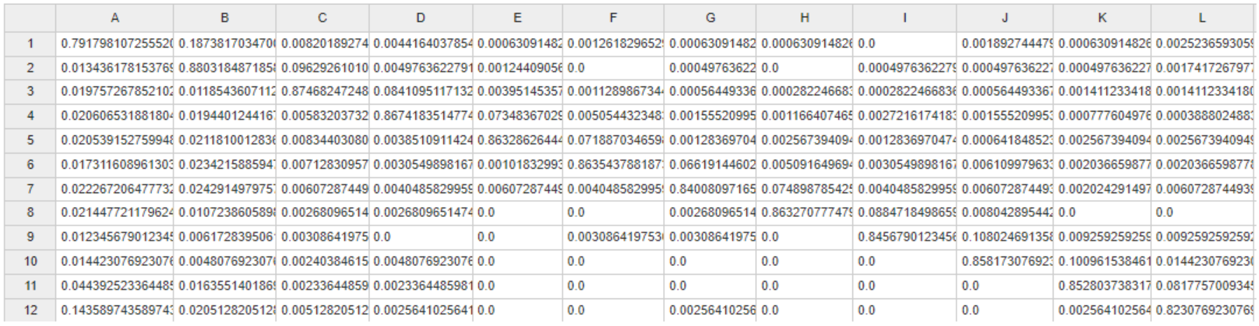
After data processing step, it comes to data relationship computation and compute the different probabilities(start, transition and emissions).
First, we calculate the start probability distribution which is the occurrence of each hidden state see figure 3.6.



	A	B	C	D	E	F
1	H1	H2	H3	H4	H5	H6
2	0.09499550494456098	0.24087503745879532	0.21234641893916692	0.15415043452202576	0.09337728498651483	0.05885525921486365
3	H7	H8	H9	H10	H11	H12
4	0.02960743182499251	0.02235540905004495	0.01941863949655379	0.024932574168414743	0.025651783038657477	0.02343422235540905

Figure 3.6: Start probability for 12 hidden state

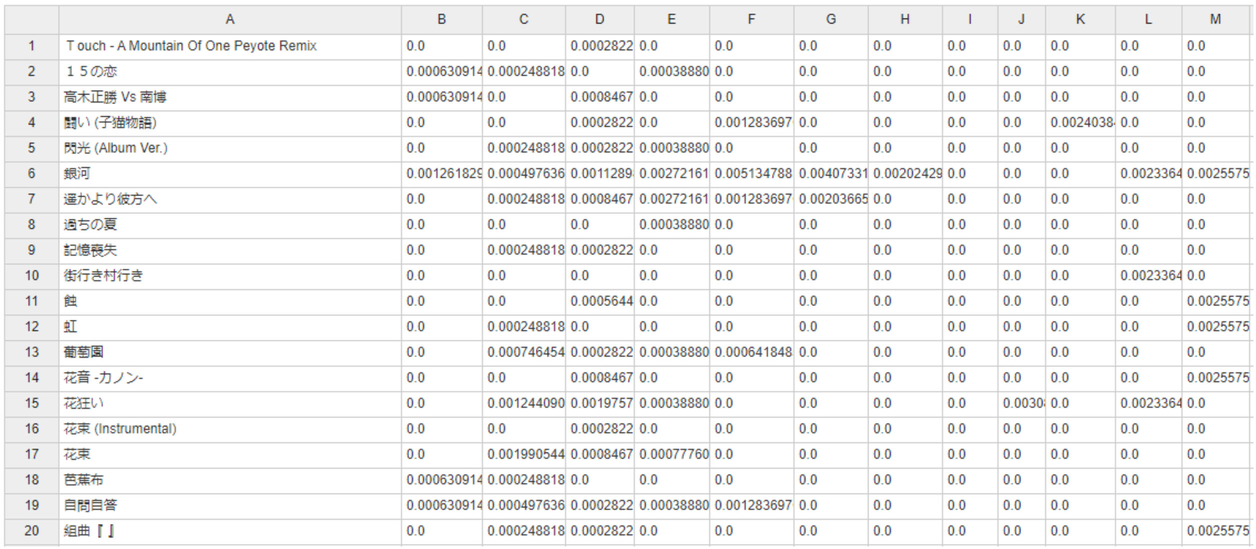
Secondly, calculate the transition probability which is the probabilities between states (the probability of navigating from state i to state j) see figure 3.7.



	A	B	C	D	E	F	G	H	I	J	K	L
1	0.79179810725552	0.187381703470	0.00820189274	0.0044164037854	0.00063091482	0.00126182965	0.00063091482	0.00063091482	0.0	0.00189274447	0.00063091482	0.00252365930
2	0.01343617815376	0.88031848718	0.09629261010	0.0049763622791	0.0012440905	0.0	0.00049763622	0.0	0.000497636227	0.00049763622	0.00049763622	0.00174172679
3	0.01975726785210	0.0118543607112	0.87468247248	0.0841095117132	0.00395145357	0.00112898673	0.00056449336	0.00028224668	0.000282246683	0.00056449336	0.001411233418	0.00141123341
4	0.02060653188180	0.01944012441	0.00583203732	0.8674183514774	0.07348367029	0.00505443234	0.00155520995	0.00116640746	0.00272161741	0.00155520995	0.00077760497	0.00038880248
5	0.02053915275994	0.02118100128	0.00834403080	0.0038510911424	0.86328626444	0.07188703465	0.00128369704	0.00256739409	0.00128369704	0.00064184852	0.002567394094	0.00256739409
6	0.01731160896130	0.02342158859	0.00712830957	0.0030549898167	0.00101832993	0.86354378818	0.06619144602	0.00509164969	0.00305498981	0.00610997963	0.002036659877	0.00203665987
7	0.02226720647773	0.02429149797	0.00607287449	0.0040485829959	0.00607287449	0.00404858299	0.84008097165	0.07489878542	0.00404858299	0.00607287449	0.002024291497	0.00607287449
8	0.02144772117962	0.01072386058	0.00268096514	0.0026809651474	0.0	0.0	0.00268096514	0.86327077747	0.08847184986	0.00804289544	0.0	0.0
9	0.01234567901234	0.00617283950	0.00308641975	0.0	0.0	0.00308641975	0.00308641975	0.0	0.84567901234	0.10802469135	0.00925925925	0.00925925925
10	0.01442307692307	0.00480769230	0.00240384615	0.004807692307	0.0	0.0	0.0	0.0	0.0	0.85817307692	0.100961538461	0.01442307692
11	0.04439252336448	0.01635514018	0.00233644859	0.0023364485981	0.0	0.0	0.0	0.0	0.0	0.0	0.852803738317	0.08177570093
12	0.14358974358974	0.02051282051	0.00512820512	0.0025641025641	0.0	0.0	0.00256410256	0.0	0.0	0.0	0.002564102564	0.82307692307

Figure 3.7: Transition probability for 12 hidden state

Lastly calculate the emission probability, the probability of each observation with each state see figure 3.8, and then assign each of this values to the HMM.



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Touch - A Mountain Of One Peyote Remix	0.0	0.0	0.0002822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	１５の恋	0.000630914	0.000248818	0.0	0.00038880	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	高木正勝 Vs 南博	0.000630914	0.0	0.0008467	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	闘い (子猫物語)	0.0	0.0	0.0002822	0.0	0.001283697	0.0	0.0	0.0	0.0	0.0024038	0.0	0.0
5	閃光 (Album Ver.)	0.0	0.000248818	0.0002822	0.00038880	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	銀河	0.00126182	0.000497636	0.0011289	0.00272161	0.005134788	0.00407331	0.00202429	0.0	0.0	0.0	0.0023364	0.0025575
7	遥かより彼方へ	0.0	0.000248818	0.0008467	0.00272161	0.001283697	0.00203665	0.0	0.0	0.0	0.0	0.0	0.0
8	過ちの夏	0.0	0.0	0.0	0.00038880	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	記憶喪失	0.0	0.000248818	0.0002822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
10	街行き村行き	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0023364	0.0
11	蝕	0.0	0.0	0.0005644	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0025575
12	虹	0.0	0.000248818	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0025575
13	葡萄園	0.0	0.000746454	0.0002822	0.00038880	0.000641848	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	花音 -カノン-	0.0	0.0	0.0008467	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0025575
15	花狂い	0.0	0.001244090	0.0019757	0.00038880	0.0	0.0	0.0	0.0	0.0030	0.0	0.0023364	0.0
16	花束 (Instrumental)	0.0	0.0	0.0002822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
17	花束	0.0	0.001990544	0.0008467	0.00077760	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
18	芭蕉布	0.000630914	0.000248818	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
19	自問自答	0.000630914	0.000497636	0.0002822	0.00038880	0.001283697	0.0	0.0	0.0	0.0	0.0	0.0	0.0
20	組曲『』	0.0	0.000248818	0.0002822	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0025575

Figure 3.8: Emission probability for 12 hidden state

2.3.3 Next-Item Recommendation

Algorithm 3: Next-Item Recommendation

```
1: extract the most likely hidden states sequence based on the observation sequence.
2: Predict next hidden state $q_{t+1}$ using the transition probability.
3: Recommend Next-Item using the emission probability.
4: Evaluate the HMM.
```

Given an observation sequence using the *Viterbie*[4] algorithm we predict the most likely hidden states sequence (which corresponds to the given sequence) and it's probability, then we predict the next hidden state using the HMM parameters, based on the last hidden state and the transition probabilities.

Then calculate the equation $o_{t+1} = \arg\max_i p(i|q_{t+1}).p(q_{t+1}|q_t)$ which means the insertion of the high-order MC, the probability of the next-item depends on all items in the sequence, and sort the output and recommend *Top-K* item on the input sequence, to make sure our model work properly it must go by the evaluation step using the evaluation metrics on the test sequence.

2.4 Implementation

This section will introduce the softwares, tools and Hardware that were used in this work.

2.4.1 Software

The implementation was done in *Python 3.6.12* with the *The Jupiter Notebook* and *Hidden Markov Model* were implemented with *Pomegranate.*

2.4.1.1 Python

Python[5] is a widely-used, interpreted, object-oriented, and high-level programming language with dynamic semantics, used for general-purpose programming (figure 3.9). It was created by *Guido van Rossum*, and first released on *February 20, 1991.* It can be used to create approximately any kind of software (Web and Internet development, Scientific and numeric computing, Desktop GUIs, Software Development, and many more). Over the last few years the open source community has developed increasingly-sophisticated data manipulation, statistical analysis, and Machine learning libraries for Python *(Pandas, NumPy, Matplotlib...)*, which has added to its popularity. There are many reasons why Python is one of the most used programming languages for data science, including: *Speed, Availability, Design goal(intuitive, easy to understand and easy to obtain).*

[4]pomegranate.readthedocs.io/en/latest/HiddenMarkovModel.viterbi.html?highlight=viterbipomegranate.hmm
[5]python.org

Figure 3.9: python programming langague logo

2.4.1.2 Pomegranate

Pomegranate[6] (figure 3.10) is a Python package that implements fast and flexible probabilistic models ranging from individual probability distributions to compositional models such as Bayesian networks and Hidden Markov Models. The core philosophy behind pomegranate is that all probabilistic models can be viewed as a probability distribution in that they all yield probability estimates for samples and can be updated given samples and their associated weights. The primary consequence of this view is that the components that are implemented in pomegranate can be stacked more flexibly than other packages.

In addition to a variety of probability distributions and models, pomegranate has a variety of built-in features that are implemented for all of the models. These include different training strategies such as semi-supervised learning, learning with missing values, and mini-batch learning.

Figure 3.10: Pomegranate logo

2.4.1.3 Anaconda Navigator

Anaconda[7] (figure 3.11) is a free and open source distribution of *Python* and *R* programming languages. with an amazing collection pre-installed data science and Machine learning packages, tools, resources, and IDEs,

Anaconda Navigator is a desktop graphical user interface included in Anaconda distribution that allows to launch applications and easily manage Conda packages, environments, and channels without using command-line commands. Our work has done with The Jupyter Notebook.

[6]pomegranate.readthedocs.io/en/latest
[7]docs.anaconda.com/

Figure 3.11: Anaconda Navigator logo

2.4.1.3.1 The Jupiter Notebook

Figure 3.12: Jupiter Navigator logo

The Jupiter Notebook[8] is an interactive computing environment[9] (figure 3.12) that enables users to author notebook documents that include (Live code, Interactive widgets, Plots, Narrative text, Equations, Images and Videos. These documents provide a complete and self-contained record of a computation that can be converted to various formats and shared with others.
The Jupyter Notebook combines three components:

1. *The notebook web application:* An interactive web application for writing and running code interactively and authoring notebook documents.

2. *Kernels:* Separate processes started by the notebook web application that runs user's code in a given language and returns output back to the notebook web application.

[8]jupyter.org/
[9]jupyter-notebook.readthedocs.io/en/stable/notebook.html

3. *Notebook documents:* Self-contained documents that contain a representation of all content visible in the notebook web application, including inputs and outputs of the computations, narrative text, equations...

2.4.2 Hardware

Our work was performed using a *DELL* personal computer with the following hardware configuration:

- Model: DELL Latitude-E5470.
- Processor: Intel® Core™ i5-6440HQ CPU @ 2.60GHz × 4.
- Random Access Memory: 8 GiB DDR4.
- Graphics: Mesa Intel® HD Graphics 530 (SKL GT2).
- Storage Capacity: 128Go SSD.
- OS name: Ubuntu 20.04.2 LTS, 64-bit.

2.5 Experiments

2.5.1 Evaluation Metrics

The performances of the proposed model is evaluated by *precision* and *recall*. These two measures are among the common metrics used for online evaluation of recommendation algorithms where the binary type of feedback is available [37].
The precision and recall are computed as:

$$Precision = \frac{TP}{TP + FP} \tag{3.1}$$

$$recall = \frac{TP}{TP + FN} \tag{3.2}$$

Where TP represents the number of correct recommendations, and FP indicate the number of incorrect recommendations. FN indicates the number of relevant items that are not included in the recommendation list [37]. Another commonly used measure is F -measure that is defined as:

$$F - measure = \frac{2.Precision.Recall}{Precision + Recall} \tag{3.3}$$

2.6 Results

This section shows the results of the proposed Hidden Markov Model on the Last.fm dataset. Table 3.2 set forth the results of the 12 hidden stat model and the table 3.3 shows the 24 hidden state model results with the application of several parameters (number of observation and user-track sequences), it is permissible to say that each change of HMM parameter is considered as a model in it self.

12 Hidden states				
Observations	Sequences	Precision	Recall	F-Mesure
3092	200	0.0368	0.2015	0.0623
11038	925	0.0173	0.0930	0.0292
14837	1168	0.0140	0.0756	0.0236

Table 3.2: Hidden Markov Model results with 12 hidden states

24 Hidden states				
Observations	Sequences	Precision	Recall	F-Mesure
3092	200	0.0175	0.0949	0.0296
11038	925	0.0117	0.0634	0.0198
14837	1168	0.0096	0.0521	0.0163

Table 3.3: Hidden Markov Model results with 24 hidden states

2.7 Discussion

Several architectures were examined from where observations, sequences and hidden states number and a 12 hidden states of HMM with the smallest number of observations was found to be the best performer depending on Precision and the F-Mesure values but with a higher value in recall. adding additional observations or hidden states always resulted in worse performances, due to the fact that precision and recall yielded a decrease in their values.

3 Conclusion

This chapter focuses on the use of one of the Machine learning models: HMM on Sequence-Aware Recommender Systems, which are becoming one of the most important recommendation approaches in practice for many domains, the proposed model for SARS submitted with different set of parameters that allows us to understand how to deal with HMM and extract the best behavior in terms of the three evaluation metrics introduced: precision, recall and f-mesure.

From that we conclude that the HMM works better with small amount of hidden states and observations or at least the gap between the observations and the hidden states number is not big, similarly the more the number of hidden states increases the more the results are unsatisfactory because it causes the emission probabilities to be less balanced (i.e. each observable won't have relations with many hidden states).

General conclusion

Summary

First, Recommender systems reported in Chapter 1 were surveyed to identify the key challenges of recommendations in most of the fields, such as the absence of users' profiles and the very short lifetime of users' interests. Most users are anonymous. Based on this thesis, Sequence-Aware Recommender Systems were proposed and their position was clarified among other Recommender systems. Also, a categorization was given, and different approaches of SARS were explained.

In Chapter 2, Machine learning was investigated in depth from developing a learning machine, types and to challenges, for the purpose of adopting an approach which is Hidden Markov Model for the development of Sequence-Aware Recommender Systems.

To answer the research question, in chapter 3 a Sequence-Aware Recommender Systems was designed, tested and evaluated an Hidden Markov Model with multiple parameters. The thesis is concluded by reflecting on the research question:

How can Hidden Markov Model be used in Sequence-Aware Recommender Systems development?

Directions for future research

The Hidden Markov Model architecture used in this thesis is relatively simple. It could be further extended by adding more advanced techniques if possible, that could further improve the predictive performance of the model.However, several points still need to be addressed for the improvement of the recommendation, such as:

- Adding the re-estimation step for taking new observabels into count automatically.
- Re-build the model with a different set of datasets in other domain such as e-commerce for the Strengthening the meaning of the given results.
- Compare the model to a set of commonly used baselines.
- Considering more contextual factors that may affect a user's choice.

Bibliography

[1] Charu C. Aggarwal,2016, Recommender Systems,NY, USA,IBM T.J. Watson Research Center

[2] D. M. Fleder and K. Hosanagar. Recommender systems and their impact on sales diversity. ACM Conference on Electronic Commerce, pp. 192–199, 2007.

[3] N. Good, J. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl.Combining collaborative filtering with personal agents for better recommendations.National Conference on Artificial Intelligence (AAAI/IAAI), pp. 439–446, 1999.

[4] J. Schafer, J. Konstan, and J. Riedl. Recommender systems in e-commerce. ACM Conference on Electronic Commerce, pp. 158–166, 1999.

[5] Luis M. de Campos, Juan M. Fernández-Luna, Juan F. Huete, Miguel A. Rueda-Morales.2010.Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks.Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S.I. Informática y de Telecomunicación, CITIC-UGR Universidad de Granada,C.P, 18071 Granada, Spain

[6] Shah Khusro, Zafar Ali, and Irfan Ullah. Recommender Systems: Issues, Challenges, and Research Opportunities, volume Volume 376, pages Pages 1179-1189. Springer, Singapore, 2016.

[7] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749.

[8] R. Burke, Hybrid recommender systems: survey and experiments, User Modeling and User-Adapted Interaction 12 (4) (2002) 331–370

[9] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Recommender systems handbook, pages 73–105.Springer, 2011. 12

[10] Robin Burke D. Knowledge-based recommender systems. Encyclopedia of library and infor- mation systems, 69(Supplement 32):175–186, 2000. 11

[11] R. Guttman, A. Moukas, and P. Maes. Agent-mediated electronic commerce: A survey, Knowledge Engineering Review, 13(2), pp. 147 159, 1998.

[12] Bruce Krulwich. Lifestyle finder: Intelligent user profiling using large-scale demographic data. AI magazine, 18(2):37–37, 1997. 11

[13] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D. and Sartin, M.: 1999, 'Combining Content-Based and Collaborative Filters in an Online Newspaper'. SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA. ¡URL: http://www.cs.umbc.edu/ian/sigir99-rec/papers/claypoolm.ps.gz¿

[14] Tran, T. and Cohen, R.: 2000, 'Hybrid Recommender Systems for Electronic Commerce'. In Knowledge-Based ElectronicMarkets, Papers from the AAAI Workshop, AAAI Technical Report WS-00-04. pp. 78 83. Menlo Park, CA: AAAI Press.

[15] Smyth, B. and Cotter, P.: 2000, 'A Personalized TV Listings Service for the Digital TV Age'. Knowledge-Based Systems 13: 53 59.

[16] Basu, C., Hirsh, H. and Cohen W.: 1998, 'Recommendation as Classfication: Using Social and Content-Based Information in Recommendation'. In: Proceedings of the 15th National Conference on Artficial Intelligence, Madison, WI, pp. 714 720.

[17] Burke, Robin. (2000). Integrating Knowledge-based and Collaborative-filtering Recommender Systems.

[18] Mooney, R. J. and Roy, L.: 1999, 'Content-Based Book Recommending Using Learning for Text Categorization'. SIGIR '99 Workshop on Recommender Systems: Algorithms and Evaluation. Berkeley, CA.

[19] Balabanovic, M.: 1998, 'Exploring versus Exploiting when Learning User Models for Text Representation'. User Modelingand User-Adapted Interaction 8(1 2), 71 102.

[20] Mobasher, B., Burke, R., Bhaumik, R., Sandvig, J.J.: Attacks and remedies in collaborative recommendation. IEEE Intelligent Systems 22, 56–63 (2007)

[21] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. ACM Trans. Inf. Syst. 39, 1, Article 10 (November 2020), 42 pages.

[22] Rendle, Steffen and Freudenthaler, Christoph and Schmidt-Thieme, Lars. (2010). Factorizing personalized Markov chains for next-basket recommendation. Proceedings of the 19th International Conference on World Wide Web, WWW '10. 811-820. 10.1145/1772690.1772773.

[23] Shoujin Wang, Liang Hu , Yan Wang , Longbing Cao, Quan Z. Sheng and Mehmet Orgun . 2019 .Sequential Recommender Systems: Challenges, Progress and Prospects. Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI-19)

[24] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. ACM Comput. Surv. 1, 1, Article 1 (February 2018), 35 pages.

[25] Massimo Quadrana,ALGORITHMS FOR SEQUENCE-AWARE RECOMMENDER SYSTEMS,POLITECNICO DI MILANO DEIB DOCTORAL PROGRAM IN INFORMATION TECHNOLOGY,2017,206 pages,consulted the 13th juin 2021

[26] Geron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems. Sebastopol, CA: O'Reilly Media. ISBN: 978-1491962299

[27] Khanna, Rahul amd Awad, Mariette. (2015). Efficient Learning Machines: Theories, Concepts, and Applications for Engineers and System Designers. 10.1007/978-1-4302-5990-9.

[28] A, ishath and K, Chaithra and B, Nishmitha and Pallavi, P and S, Raghavendra and Prasanna, Mahesh. (2019). Survey on Artificial Intelligence. International Journal of Computer Sciences and Engineering. 7. 1778-1790. 10.26438/ijcse/v7i5.17781790.

[29] Sharma, Shree Krishna and Wang, Xianbin. (2018). Towards Massive Machine Type Communications in Ultra-Dense Cellular IoT Networks: Current Issues and Machine Learning-Assisted Solutions.

[30] F. Yang, S. Balakrishnan and M. J. Wainwright, "Statistical and computational guarantees for the Baum-Welch algorithm," 2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton), 2015, pp. 658-665, doi: 10.1109/ALLERTON.2015.7447067.

[31] Awad, Mariette and Khanna, Rahul. (2015). Hidden Markov Model. 10.1007/978-1-4302-5990-9-5.

[32] Dymarski, P. (2011). Hidden Markov models: Theory and applications. Rijeka, Croatia: InTech.

[33] Daniel Jurafsky and James H. Martin. 2009. Speech and Language Processing (2nd Edition). Prentice-Hall, Inc., USA.

[34] Nilsson, M. (2005). First Order Hidden Markov Model : Theory and Implementation Issues.

[35] Athanasopoulou, E. and Hadjicostis, Christoforos. (2003). Upper and lower bounds on FSM switching activity. 5. V-417 . 10.1109/ISCAS.2003.1206301.

[36] Coelho, J. P., Pinho, T. M., and Boaventura-Cunha, J. (2019). Hidden Markov models: Theory and implementation using Matlab.

[37] Hosseinzadeh Aghdam, Mehdi. (2018). Context-aware recommender systems using hierarchical hidden Markov model. Physica A: Statistical Mechanics and its Applications. 518. 10.1016/j.physa.2018.11.037.